Parsimonious Principles of Deep Neural Networks

by

Minyoung Huh

B.S., University of California, Berkeley, 2016
M.S., Massachusetts Institute of Technology, 2022

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2024

Authored by:	Minyoung Huh Department of Electrical Engineering and Computer Science August 31, 2024
Certified by:	Phillip Isola Professor of EECS, Thesis Supervisor
Certified by:	Pulkit Agrawal Professor of EECS, Thesis Supervisor
Accepted by:	Leslie A. Kolodziejski Professor of Electrical Engineering and Computer Science Chair, Department Committee for Graduate Students

Parsimonious Principles of Deep Neural Networks

by

Minyoung Huh

Submitted to the Department of Electrical Engineering and Computer Science
on August 31, 2024 in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

ABSTRACT

At the core of human intelligence lies an insatiable drive to uncover the simple underlying principles that govern the world's complexities. This quest for parsimony is not unique to biological cognition but also seems to be a fundamental characteristic of artificial intelligence systems. In this thesis, we explore the intrinsic simplicity bias exhibited by deep neural networks — the powerhouse of modern AI. By analyzing the effective rank of the learned representation kernels, we unveil the observation that these models have an inherent preference for learning parsimonious relationships in the data. We provide further experimental results to support the hypothesis that simplicity bias is a good inductive bias for finding generalizing solutions. Building upon this finding, we present the Platonic Representation Hypothesis — the idea that as AI systems continue to grow in capability, they will converge toward not only simple representational kernels but also a common one. This phenomenon is evidenced by the increasing similarity of models across domains, suggesting the existence of a Platonic "ideal" way to represent the world. However, this path to the Platonic representation necessitates scaling up AI models, which poses significant challenges regarding computational demand. To address this obstacle, we conclude the thesis by proposing a framework for training a model with parallel low-rank updates to effectively reach this convergent endpoint.

Thesis supervisor: Phillip Isola

Title: Professor of EECS

Thesis supervisor: Pulkit Agrawal

Title: Professor of EECS

Acknowledgments

This thesis would not have been possible without the support of those who have been beside me throughout this journey.

First and foremost, I would like to thank my advisors: Phillip Isola and Pulkit Agrawal, and my undergraduate advisor, Alyosha Efros. Pulkit introduced me to deep learning research and provided immense support when I was an undergraduate researcher at Berkeley. Through him, I met Alyosha, who instilled in me the mantra "data is everything." Alyosha's classes in computational photography, machine learning, and computer vision ignited my passion for these fields. Both Alyosha and Pulkit encouraged me to pursue a doctoral degree, a decision that has enriched my life in countless ways. Pulkit, you have been a great advisor and a friend. You have always had my best interests at heart, and I will always be thankful.

During my time in Efros' group, I met Phillip Isola, whose GAN tutorials and fascinating perspectives on artificial intelligence inspired me throughout my undergraduate studies. His work in representation learning and generative modeling was a major reason I pivoted from computer vision to machine learning during my graduate studies. Phillip, your patience and humble advice on academic thinking, as well as on how to be a good researcher and person, have profoundly shaped who I am today.

I would also like to thank my thesis committee member, Yoon Kim, who provided fruitful discussions on both research and life throughout my PhD. I also want to thank his students and the supportive environment they fostered during my last two years .

I am also grateful to my intern mentors: Ning Zhang, Aaron Hertzmann, Richard Zhang, Jun-Yan Zhu, Sylvain Paris, Max Vladymrov, Ben Poole, and Luke Metz. Thank you for the opportunities and encouragement to explore some wild ideas, even if the projects did not always come to fruition.

I would like to thank my collaborators: Toru Lin, Jingwei Ma, Jyo Pari, Brian Cheung, Tongzhou Wang, Uzay Girit, Hyojin Bahng, Andrew Liu, Shao-Hua Sun, Yang Liu, Andrew Owens, Jeremy Bernstein, Tim Large, Hossein Mobahi, and Antonio Torralba. Working with you has been invaluable to my career.

Special thanks go to the graduate students at BAIR who welcomed and supported me as an undergraduate:

- Jitendra's group: Saurabh Gupta, Weicheng Kuo, Shubham Tulsiani, Ke Li, Abhishek Kar, Georgia Gkioxari, David Fouhey, and Angjoo Kanazawa.

- Efros' group: Deepak Pathak, Jun-Yan Zhu, Richard Zhang, Tinghui Zhou, Shiry Ginosar, Taesung Park, and Andrew Owen.
- Trevor's group: Evan Shelhamer and Jeff Donahue.

More than they realize, I have gained invaluable knowledge through osmosis.

My experience at MIT and Berkeley would not have been complete without the friends I've made along the way. I'd like to thank my dearest friends Yen-Chen Lin, Lucy Chai, Wei-Chiu Ma, Tongzhou Wang, Ching-Yao Chuang, Yonglong Tian, and Caroline Chan for making my time at MIT truly memorable. I'm also grateful to Dave Epstein for the long interview grinding sessions, Allan Jabri for moral support during the interview process, and Brian Cheung for being one of the most creative collaborators I've worked with. Additional thanks to Han Guo and Jyo Pari for the fruitful research discussions and for co-organizing the ATLAS/MLScale seminars.

I would also like to acknowledge my friends, lab mates, and mentees who have been part of this journey: Tao Chen, Anurag Ajay, Ruben Castro, Anthony Simeonov, Aviv Netanyahu, Richard Li, Gabe Margolis, Idan Shenfeld, Seungwook Han, Felix Wang, Joseph Suarez, Toru Lin, Jingwei Ma, Hyojin Bahng, Shobhita Sundaram, Prafull Sharma, Ge Yang, Jeremy Bernstein, Yonglong Tian, Xavier Puig Fernandez, Chen-Hsuan Lin, David Bau, Shuang Li, Pratyush Sharma, Manel Barada, Kabir Swain, Joanna Materzynska, Adrian Rodriguez, Yu Sun, Tim Brooks, Michael Janner, Parsa Mahmoudieh, Andrew Liu, Sheng-Yu, Jasmine Collins, Sasha Sax, Ashish Kumar, Jihui Jin, Inho Cho, Clinton Wang, Angela Park, Roy Kim, Luna, Merissa Chen, Marvin Do, Jeannie Chung, Michael Lee, Ryan Gu, Thomas Kil, Daniel Cho, Peter Kim, David Paek, Edward Kil, Sahaana Suri, Jene Li, Jenny Hsieh, Dima Smirnov, Andi Peng, Uzay Girit, Kevin Frans, Stephanie Fu, Akarsh Kumar, Ishaan Preetam-Chandratreya, Patrick Chen, and Victor Butoi.

Lastly, my deepest gratitude goes to my family, to whom this thesis is dedicated. To my parents, your unwavering support throughout my life has shaped who I am today. Your unconditional love and affection have been the foundation that has enabled me to pursue my career. To my brother, your curiosity in science has always inspired my desire to pursue this path, and for that, I am forever grateful. To my grandfather, my biggest fan, I hope to continue making you proud. And foremost, to my partner, your steadfast support has guided me through the toughest times during my PhD. Thank you for your patience and love. To my dogs Leo and Sally, woof woof woof (translated to "Thanks for the emotional support, I will buy you more treats.").

Contents

Chapter 1

Prologue

The pursuit of science is to uncover the simple underlying principles that govern the complexities of the universe. From birth, humans, along with many pseudo-intelligent biological systems, are driven to observe the world, discern patterns, and extrapolate these observations to maximize utility or, for the curious ones, to quench the thirst for knowledge. The essence of scientific inquiry, however, makes one crucial assumption: that patterns will *"always"* exist in our data. But why should they?

An often unspoken axiom in science is the assumption that the world and the rules that govern it are inherently simple, and it's the interaction of these simple rules that gives rise to complex processes, like the ones we see in Conway's Game of Life. This notion is perfectly summarized by Nobel Laureate Murray Gell-Mann:

> *"Regularities come from two fundamental laws of physics: the unified theory of all the particles and the forces, and the initial condition of the universe. We believe both of these are simple."*

The goal of fundamental sciences is, therefore, to infer these simple underlying rules from complex data. And because the underlying process is simple, we assume that even complex data generated from this process must contain structure.

This brings us to a long-standing interest in understanding the complexity of data and the principles that underpin it. In Algorithmic Information Theory (AIT), this concept is often referred to as the Minimum Description Length (MDL), which seeks to determine the shortest possible description that can compress the observed data without losing any information. Simply put, MDL asks, "How many instructions are needed to fully explain the patterns observed in the data?" The belief that MDL reflects the underlying data-generating process is often referred to as the *Compression Hypothesis*. This is an unofficial umbrella term used to argue for the law of parsimony (Ockham 1639; Gell-Mann 1994; Kolmogorov 1965; Solomonoff 1964a; Rissanen 1978; Schmidhuber 2007; Hutter 2006), where the simplest explanation is likely to be the correct explanation. Of course, there are counterexamples, but this idea has generally stood the test of time. This led me to ask: Is this desire to infer short

descriptions unique to humans? Or do machines, like humans, also exhibit a preference for simpler solutions? If they do, it could suggest that artificial systems have the potential to exhibit human-like generalization behaviors.

This would also explain the recent paradigm shift toward multifunctional systems in the field of AI, where larger and more capable models generalize and adapt better to a wide range of tasks. Since the underlying data-generating process (*e.g.*, the universe) is the same, intelligent systems, artificial or biological, should likely arrive at the same understanding of the relationships present in the data (Cao and Yamins 2021; Bansal, Nakkiran, and Barak 2021). The convergence of AI systems toward more homogeneous capabilities raises several critical questions: What drives this convergence? Is it likely to continue? What is its ultimate trajectory? How can we efficiently make progress toward more unified AI systems? This thesis aims to provide insights into these aspects and is structured as follows:

Chapter 2: Explores deep neural networks' inherent preference for simple relationships. We examine this aspect by studying the effective rank of the kernels generated by the model.

Chapter 3: Introduces the convergence hypothesis, offering both empirical and theoretical evidence that AI systems are moving toward a common representational endpoint. We also argue that scaling is essential for achieving this convergence.

Chapter 4: Demonstrates an efficient approach to scaling model training by introducing a distributed framework that uses parallel low-rank adapters to collectively train large neural networks.

Chapter 5: Summarizes findings and provides concluding thoughts on these research directions.

Chapter 2

The low-rank simplicitiy bias of deep neural networks

Deep neural networks have demonstrated remarkable generalization capabilities, but a fundamental question remains: do these models generalize because they inherently find simple solutions? In this chapter, we explore the intrinsic characteristics of neural networks that determine the complexity of the solutions these models tend to gravitate toward. Specifically, we investigate the concept of simplicity by examining the effective rank of the kernels produced by these models — the number of bases used to characterize the learned relationships in the data.

Measuring simplicity in neural networks can be approached in various ways, and analyzing the rank of kernels is just one method. However, it provides valuable insights into the complexity — or lack thereof — of the solutions that neural networks converge upon. While factors such as data, optimizer, and architecture all influence a model's tendency toward simplicity, our focus here is on the architectural bias itself. We argue that the architecture of deep neural networks inherently promotes a preference for low-rank, and therefore simpler, solutions.

2.1 Introduction

It has become conventional wisdom that the more layers one adds, the better a deep neural network (DNN) performs. This guideline is supported, in part, by theoretical results showing that deeper networks can require far fewer parameters than shallower networks to obtain the same modeling "capacity" (Eldan and Shamir 2016). While it is not surprising that deeper networks are more expressive than shallower networks, the fact that state-of-the-art deep networks do not overfit, despite being heavily over-parameterized, defies classical statistical theory (Geman, Bienenstock, and Doursat 1992; Zhang, Bengio, Hardt, et al. 2021; Belkin, Hsu, Ma, et al. 2019) – e.g., (Dosovitskiy, Beyer, Kolesnikov, et al. 2020) trains a 632 million parameter, 200+ layer model, on 1.3 million images.

The belief that *over-parameterization via depth improves generalization* is used axiomatically in the design of neural networks. Unlike conventional regularization methods that penalize

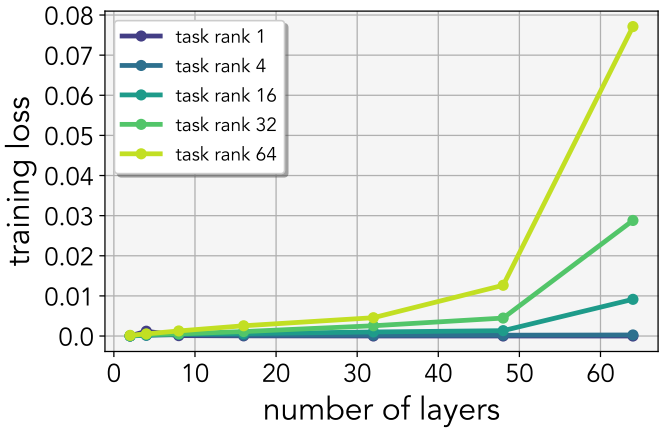


Figure 2.1: **Deep nets *struggle* to fit high-rank *linear* functions:** We report the training loss of neural networks of different depths optimized to solve linear regression. The rank of the underlying linear function is varied in the range $[1, 64]$. While shallow networks achieve zero training loss, the training loss worsens with increased depth and task rank (see Appendix 7.5 for training details).

model complexity (e.g., ℓ_1/ℓ_2 penalty), over-parameterization does not. Yet, like explicit regularization, over-parameterization appears to prevent the model from over-fitting (Belkin, Hsu, Ma, et al. 2018; Nakkiran, Kaplun, Bansal, et al. 2019). Why this *implicit regularization* works is still an ongoing area of research.

In this work we made a rather unexpected discovery that provides clues into why and when *over-parameterization* could help, which echoes a lot of the prior works on the spectral bias of deep networks. This discovery is the result of the following experiment: training ReLU networks with varying depths on a set of linear regression tasks $Y = W^*X$. For some randomly sampled X we minimize the least-squares error between the prediction $\hat{Y}$ and the ground-truth targets Y. In Figure 2.1, we plot the converged loss when varying the depth of model and the underlying rank of the task: $\text{rank}(W^*) = \{1, 4, 16, 32, 64\}$. The results reveal that deeper networks touted for their ability to model complex functions struggle to fit even (high-rank) linear functions. In contrast, shallower networks perfectly minimize the loss.

One explanation of these results is improper optimization of neural network parameters. We used standard SGD based optimizers and experimented with a wide range of hyper-parameters that we detail in Appendix 7.5 along with the code in the supplementary materials. While there may exist an optimization algorithm that can perfectly minimize training error, we do not know of such an optimizer. At first, our result might seem to be at odds with the work of (Zhang, Bengio, Hardt, et al. 2021) observing that deep networks (8 layers) can achieve zero training error on random data. However, our results are consistent because (Zhang, Bengio, Hardt, et al. 2021) did not experiment with deeper networks, and predicting labels from images is (loosely speaking) not a full rank prediction problem.

The second possibility is our hypothesis that *deep over-parameterized networks are biased to find low-rank solutions.* Results in Figure 2.1 corroborate this hypothesis, but the problem is

that the concept of rank is not defined for a non-linear network. However, it is still possible to study the rank of the feature embeddings learned by the penultimate layer of the neural network. In the case of a linear neural network, the embedding and parameter rank are equivalent. In the remainder of this work, we probe the relationship between embedding rank and depth. Our findings indeed strengthen the hypothesis that deeper networks find lower (embedding) rank solutions.

Prior work has shown that over-parameterized linear networks find minimum norm solutions (Gunasekar, Woodworth, Bhojanapalli, et al. 2017; Arora, Cohen, Hu, et al. 2019), which in special cases, is equivalent to finding low-rank solutions. Past work has also suggested that over-parameterized *non-linear* networks find "simple" solutions (Valle-Perez, Camargo, and Louis 2019) but using measures of simplicity other than rank. Our work ties together these two lines of research to analyze the relationship between *embedding rank* and depth in *non-linear* networks. The fact that deeper networks are primed to learn solutions that have low-embedding rank may also explain why they generalize despite being over-parameterized – most natural data (e.g., images) actually lies on a low-dimensional manifold, and common problems such as classification require predicting quantities that are much lower-dimensional than the inputs.

2.2 Related works

Linear networks Linear networks have been used in lieu of non-linear networks for analyzing the generalization capabilities of deep nets. These networks have been widely used for analyzing learning dynamics (Saxe, Mcclelland, and Ganguli 2014) and forming generalization bounds (Advani, Saxe, and Sompolinsky 2020). Notable work from (Arora, Cohen, and Hazan 2018) shows that over-parameterization induces training acceleration which cannot be approximated by an explicit regularizer. Furthermore, (Gunasekar, Woodworth, Bhojanapalli, et al. 2017) shows that linear models with gradient descent converge to a minimum nuclear norm solution on matrix factorization. More recently, (Li, Luo, and Lyu 2020) demonstrated that gradient descent acts as a greedy rank minimizer in matrix factorization, and (Bartlett, Long, Lugosi, et al. 2020; Bartlett, Montanari, and Rakhlin 2021) argues that gradient descent in over-parameterized models leads to benign overfitting. Although mainly used for simplifying theory, (Bell-Kligler, Shocher, and Irani 2019) demonstrate the practical applications of deep linear networks.

Low-rank bias Deep linear neural networks have been known to be biased towards low-rank solutions. One of the most widely studied regimes is on matrix factorization with gradient descent under isometric assumptions (Tu, Boczar, Simchowitz, et al. 2016; Ma, Wang, Chi, et al. 2018; Li, Ma, and Zhang 2018), and further studied on least-squares (Gidel, Bach, and Lacoste-Julien 2019). (Arora, Cohen, Hu, et al. 2019) showed that matrix factorization tends to low nuclear-norm solutions with singular values decaying faster in deeper networks. Note that these works focus on why gradient descent finds low-rank solutions. (Pennington, S. Schoenholz, and Ganguli 2018) showed that the spectral distribution of the input-output Jacobian is determined by depth. For non-linear networks, understanding the biases has been mostly empirical, with the common theme that over-parameterization of depth or width

improves generalization (Neyshabur, Tomioka, and Srebro 2015; Nichani, Radhakrishnan, and Uhler 2020; Golubeva, Neyshabur, and Gur-Ari 2021; Hestness, Narang, Ardalani, et al. 2017; Kaplan, McCandlish, Henighan, et al. 2020). These aforementioned theories have also been adopted for auto-encoding (Jing, Zbontar, et al. 2020) and model compression, (Guo, Alvarez, and Salzmann 2020). More recently, (Pezeshki, Kaba, Bengio, et al. 2020) have observed that SGD learns to capture statistically dominant features, which leads to learning low-rank solutions, and (Baratin, George, Laurent, et al. 2021) observed that the alignment of the features acts as an implicit regularizer during training.

Simplicity bias Recent work has indicated that gradient descent in linear models finds max-margin solutions (Soudry, Hoffer, Nacson, et al. 2018; Nacson, Lee, Gunasekar, et al. 2019; Gunasekar, Lee, Soudry, et al. 2018). Separately, in the perspective of algorithmic information theory, (Valle-Perez, Camargo, and Louis 2019) demonstrated that deep nets' parameter space maps to low-complexity functions. Furthermore, (Nakkiran, Kaplun, Kalimeris, et al. 2019), and (Arpit, Jastrzębski, Ballas, et al. 2017) have shown that networks learn in stages of increasing complexity. Whether these aspects of simplicity bias are desirable has been studied by (Shah, Tamuly, Raghunathan, et al. 2020).

Complexity measures A growing number of works have found matrix norm to not be a good measure for characterizing neural networks. (Shah, Kyrillidis, and Sanghavi 2018) shows that the minimum norm solution is not guaranteed to generalize well. These findings are echoed by (Razin and Cohen 2020), which demonstrates that implicit regularization cannot be characterized by norms and proposes rank as an alternative measure.

2.3 Preliminaries

Simple linear network A simple linear neural network transforms input $x \in \mathbb{R}^{n\times 1}$ to output $\hat{y} \in \mathbb{R}^{m\times 1}$, with a learnable parameter matrix $W \in \mathbb{R}^{m\times n}$,

$$\hat{y} = Wx. \tag{2.1}$$

For notational convenience, we omit the bias term.

Over-parameterized linear networks One can over-parameterize a *linear* neural network by defining d matrices $\{W_i\}_{i=1}^{d}$ and multiplying them successively with input x:

$$\hat{y} = W_d W_{d-1} \cdots W_1 x = W_e x, \tag{2.2}$$

where $W_e = \prod_{i=1}^{d} W_i$. As long as the matrices are of the correct dimensionality — matrix W_d has m columns, W_1 has n rows, and all intermediate dimensions $\{\dim(W_i)\}_{i=2}^{d-1} \geq \min(m, n)$ — then this over-parameterization expresses the same set of functions as a single-layer network. We disambiguate between the collapsed and expanded set of weights by referring to $\{W_i\}$ as the over-parameterized weights and W_e as the *end-to-end* or the *effective weights*.

Non-linear networks For *non-linear* network, activation function ψ (e.g. ReLU) is interleaved between the weights:

$$\hat{y} = W_d\psi(W_{d-1}\dots\psi(W_1(x))) \tag{2.3}$$

In contrast to linear networks, non-linear models are more expressive with more layers.

Effective rank We characterize the rank of a matrix using a continuous measure known as the *effective rank*:

Definition 2.3.1 (Effective rank)**.** (Roy and Vetterli 2007) For any matrix $A \in \mathbb{R}^{m\times n}$, the effective rank ρ is defined as the Shannon entropy of the normalized singular values:

$$\rho(A) = -\sum_{i=1}^{\min(n,m)} \bar{\sigma}_i \log(\bar{\sigma}_i),$$

where $\bar{\sigma}_i = \sigma_i / \sum_j \sigma_j$ are normalized singular values, such that $\sum_i \bar{\sigma}_i = 1$. Also referred to as the spectral entropy. Without loss of generality, we drop the exponentiation for convenience.

This measure gives us a meaningful representation of "continuous rank", which is maximized when the magnitude of the singular values are all equal and minimized when a single singular value dominates relative to others. The effective rank provides us with a metric that summarizes the distribution envelope.

Embedding maps A parameteric function $f_{\{W\}} \in \mathcal{F}_{\mathcal{W}}$ is a neural network parameterized with a set weights $\{W\} = \{W_1, \dots, W_d\}$ that maps the input space to the output space $\mathcal{X} \to \mathcal{Y}$. For a dataset of size q, the input and output data is $X \in \mathbb{R}^{n\times q}$ and $Y \in \mathbb{R}^{m\times q}$. Then, the predicted output is $\hat{Y} = W_d\psi(\Phi) = f_{\{W\}}(X)$, where $\Phi \in \mathbb{R}^{n'\times q}$ is the last-layer embedding and $W_d \in \mathbb{R}^{m\times n'}$ is the last layer of the network.

We analyze the embedding space by computing the effective rank on the Gram/kernel matrix $K \in \mathbb{R}^{p\times p}$ where p is the size of the test set. The ij-th entry of the Gram matrix corresponds to a distance kernel $K_{ij} = \kappa(\phi_i, \phi_j)$ where ϕ_i corresponds to the i-th column of Φ. We use the model's intermediate features before the linear classifier and use cosine distance kernel: $\kappa(\phi_i, \phi_j) = \frac{\phi_i \phi_j^T}{\|\phi_i\|\|\phi_j\|}$, a common method for measuring distances in feature space (Kiros, Zhu, Salakhutdinov, et al. 2015; Zhang, Isola, Efros, et al. 2018).

In natural data, it is often assumed that we are trying to discover a low-rank relationship between the input and the label. For example, a model that overfits every training sample without inferring any structure on the data will generally have a test gram-matrix that is a higher rank than that of a model that has learned parsimonious representations.

A lower rank on held-out data indicates less excess variability and is indicative of studying generalization and robustness. The intuition becomes clearer in linear networks since the rank of the Gram matrix depends on the rank of the linear transformation computed by the network. We illustrate this empirically in Appendix 7.11, where we see that there is a tight relationship between the rank of the linear weight matrix and the resulting Gram matrix.

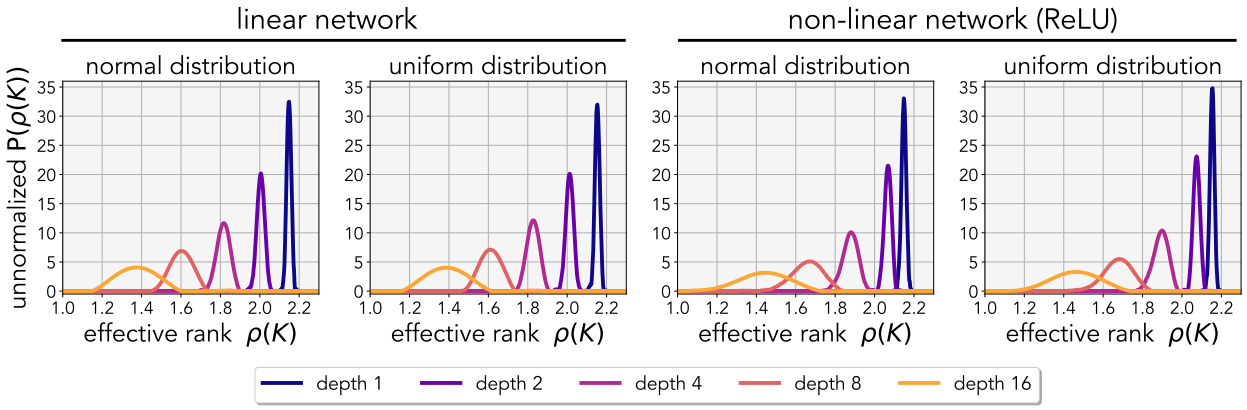


Figure 2.2: **Deep networks are biased toward low-rank:** The approximated probability density function (PDF) of the effective rank ρ over the Gram matrix is computed from features of the networks. The Gram matrix is computed with 256 random inputs, and we use 4096 network parameter samples to approximate the cumulative distribution function. The CDF is used to compute the PDF via the finite difference method. We apply (Savitzky and Golay 1964) filter to smoothen out the approximation. There exists more probability mass for lower-rank rank embeddings when adding more layers. The experiment is repeated for both normal and uniform distributions.

2.4 The parameterization bias of depth

Given that our models can always fit the data, what are the implications of searching for the solution in the over-parameterized model? In linear models, this is equivalent to searching for solutions in $\{W_i\}$ versus directly in W_e. One difference is that the gradient direction $\nabla_{\{W_i\}}\mathcal{L}(\{W_i\})$ is usually different than $\nabla_{W_e}\mathcal{L}(W_e)$ for a typical loss function $\mathcal{L}$ (see Appendix 7.9). The consequences of this difference have been previously studied in linear models by (Arora, Cohen, and Hazan 2018; Arora, Cohen, Hu, et al. 2019), where the over-parameterized update rule has been shown to accelerate training and encourage singular values to decay faster, resulting in a low nuclear-norm solution. Here we motivate a result from the perspective of parameter volume space.

Conjecture 2.4.1. Deeper networks have a greater proportion of parameter space that maps the input data to lower-rank embeddings; hence, deeper models are more likely to converge to functions that learn simpler embeddings.

We now provide a set of empirical observations that supports our conjecture. Our work and existing theoretical works on gradient descent biases are *not* mutually exclusive and are a likely complement. We emphasize that we do *not* make any claims on the simplicity of the function, but only on the simplicity – lower effective rank – of the embeddings.

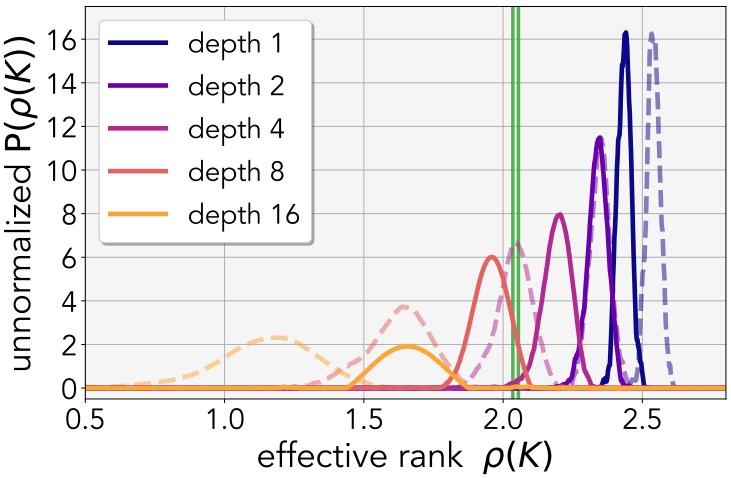


Figure 2.3: **Distribution at convergence**: Rank distribution after training the network with gradient descent. The dotted line indicates the initial distribution, the solid line indicates the converged distribution, and the green line indicates the task rank.

2.4.1 Low-rank simplicity bias of deep networks

Observation 2.4.1. Randomly initialized deep nets are biased to correspond to Gram matrices with a low effective rank.

When sampling random neural networks, both linear and non-linear, we observed that the Gram matrices computed from deeper networks have a lower effective rank. We quantify this observation by computing the distribution over the effective rank of the Gram matrix in Figure 2.2. Here, the weights of the neural networks are initialized using uniform $W_i \sim \mathcal{U}(\cdot, \cdot)$ or Normal distributions $W_i \sim \mathcal{N}(\cdot, \cdot)$. The input, output, and intermediate dimensions are 32, giving parameters $\{W_i\} \in \mathbb{R}^{d \times 32 \times 32}$ for a network with d layers. We draw 4096 random parameter samples and compute the effective rank on the resulting Gram matrix. We see that the distribution density shifts towards the left (lower effective rank) when increasing the number of layers. These distributions have small overlap and smoothen out with increased depth. This observation shows that depth correlates with lower effective rank embeddings.

The low-rank bias becomes more intuitive in linear models as there is a simple way to relate the Gram matrix to the weights of the model $K \approx (W_{d-1:1}X)^T(W_{d-1:1}X)$. Intuitively, if any constituent matrices are low-rank, then the product of matrices will also be low-rank – the product of matrices can only decrease the rank of the resulting matrix: $\text{rank}(AB) \leq \min(\text{rank}(A), \text{rank}(B))$ (Friedberg, Insel, and Spence 2003). In Appendix 7.11, we show that as the depth of the model increases, both the effective rank of the Gram matrix and the weights decrease together. Another way to interpret our observation is that for linear models, over-parameterization does not increase the expressivity of the function but re-weights the likelihood of a subset of parameters – the hypothesis class. For non-linear models, we *cannot* make the same claims.

Although uniformly sampling under the parameter distribution is an unbiased estimator of

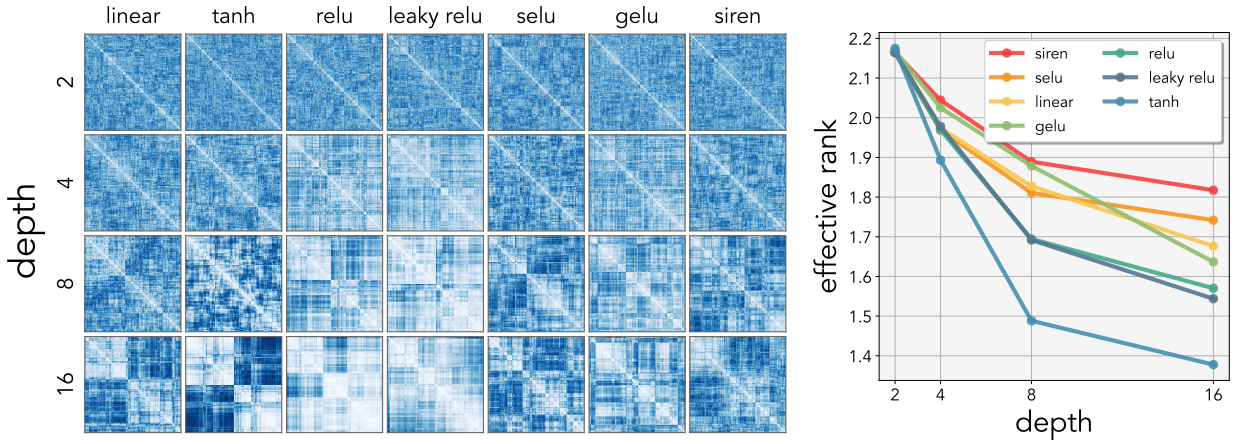


Figure 2.4: **Gram matrices of networks:** Gram matrices of neural networks trained with various non-linearities and depth. The Gram matrix is computed using the cosine-distance on the features of the test-set near zero-training loss. Increasing the number of layers decreases the effective rank of the Gram matrix on a variety of non-linear activation functions. The Gram matrix is hierarchically clustered ((Rokach and Maimon 2005)) for visualization. We observe the emergence of block structures in the Gram matrix as we increase the number of layers, indicating that the embeddings become lower rank with depth.

the volume of the parameter space, it is certainly possible that a sub-space of the parameters is more likely to be observed under gradient descent. Hence, by naively sampling networks, we may never encounter model parameters that gradient descent explores. In light of this, we repeat our experiment above by computing the PDF on randomly sampled parameters after taking n gradient descent steps.

Observation 2.4.2. Deep neural networks trained with gradient descent also learn to map data to simple embedding with low effective rank.

Figure 2.3 illustrates the change in distribution as we train our model to convergence using gradient descent. Each randomly drawn network sample is trained to minimize the least-squares error. The initial distribution is plotted with dotted lines, and the converged distribution is plotted with solid lines. As the model is trained, the distribution of the rank shifts towards the ground-truth rank (green line). Training the model with gradient descent results in a distribution that is still largely dependent on depth; this reaffirms that the role of optimization does not remove the parameterization bias in deep models. In fact, if the bias stems from the model's parameterization, the same bias must also exist under other common and natural choices of optimizers. We investigate this claim in the next section.

In Figure 2.4, we further visualize the learned Gram matrices when varying the depth of the model. The Gram matrices trained with various non-linear activation functions also emit the same low-rank simplicity bias. These activation functions include standard functions such as ReLU and Tanh as well as recently popularized non-linear functions such as GeLU ((Hendrycks and Gimpel 2016)), and the sinusoidal activation function from SIREN ((Sitzmann, Martel, Bergman, et al. 2020)). By hierarchically clustering (Rokach and Maimon 2005) these Kernels,

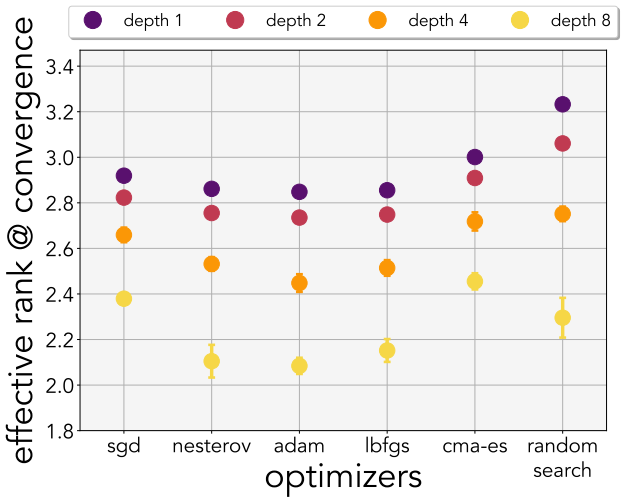


Figure 2.5: **Low-rank bias & optimizers**: Least-squares trained on linear neural networks using various optimization methods. The rank of the converged Gram matrix is correlated with the depth of the network. The experiment is repeated 5 times.

we can directly observe the emergence of block structures in the Gram matrices as we increase the number of layers, implying that the embeddings become lower rank with depth.

2.4.2 Is the low-rank bias specific to gradient descent?

Observation 2.4.3. Deep neural networks trained with common and natural choices of optimizers also exhibit the low-rank embedding bias.

The low-rank bias of deep networks has been primarily studied under the context of first-order gradient decent (Arora, Cohen, and Hazan 2018; Arora, Cohen, Hu, et al. 2019): *how and why does gradient descent converge to low nuclear norm solution*. In contrast, our conjecture focuses on the bias of parameterization of the network and *not* on the bias introduced by the gradient descent. Since parameterization bias exists regardless of the optimizer choice, we would expect to observe the low-rank simplicity bias on a wide range of optimizers. We directly show this in Figure 2.5 by ablating across various popular choices of optimizers on least-squares with linear networks. Here, we compare against Nesterov ((Nesterov 1983); momentum), ADAM ((Kingma and Ba 2015a); hessian approximator), L-BFGS ((Liu and Nocedal 1989); second-order), CMA-ES ((Hansen, Müller, and Koumoutsakos 2003); evolutionary-search), and random search. All models were trained to zero training error except for random search. For random search, we randomly initialize the network $100,000$ times and take the best performing sample. As we have seen with gradient descent, the experiment indicates that even when we train with a wide suite of commonly used optimizers, the solution obtained by these models depends on how the model was originally parameterized.

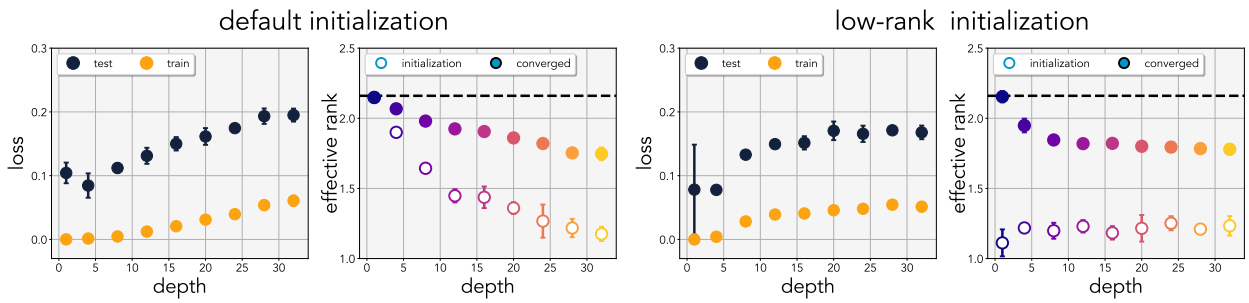


Figure 2.6: **Bias of parameterization:** The effective rank of the Gram matrix from initialization to convergence on various depth. For each depth, we train a linear network using gradient descent on least squares regression. We repeat our experiments 5 times with different seeds, and we report the median of these runs. The rank at initialization and convergence is indicated by white and colored dots, respectively. For deeper models, the effective-rank is lower at initialization because the product of normally distributed weights is no longer normally distributed. On the right, we initialize the networks with the same low-rank distribution of weights as the 32-layer model. We observe that shallower networks tend to converge to higher rank embeddings.

2.4.3 Can the bias be explained solely by initialization?

The previous set of experiments indicates that deeper networks are biased towards low-rank embedding at both initialization and convergence. In these experiments, the random settings of neural networks had different initial distributions. This happens because, even if the individual weights are normally distributed, the weights constructed from a series of matrix multiplications result in a distribution that has a high density around zero. For example, the product of 2 normally distributed weights becomes symmetric χ-squared distribution, with 1 degrees of freedom. Hence, one could argue that the converged solutions are low-rank because of the initialization bias.

Observation 2.4.4. Deep neural networks are biased towards learning low-rank embeddings and are insensitive to initialization.

To test whether the initialization of the model affects the rank of the converged solution. We optimize our network $W \in \mathbb{R}^{d\times 32\times 32}$ on least-squares where the task-rank is set to 24. All models are trained for 4000 epochs using gradient descent, and the best learning rate is chosen for each depth. In Figure 2.6 (left), for models using *default initialization*, we show that increasing the number of layers decreases the effective rank of the Gram matrix at convergence. We repeat the experiment in Figure 2.6 (right) by initializing the over-parameterized models with the distribution associated with the 32-layer linear network. Following a similar trend to that of default initialization, we observe that deeper models learn embeddings that are a lower rank than the shallower counterparts. Although initialization is not insignificant, we see that the depth of the model has tight control over the solution which the model explores. For deeper networks, the majority of the parameter volume is mapped to low-rank embedding (Observation 2.4.1), and therefore it is expected that a typical search algorithm would likely encounter parameters that map to low-rank embeddings regardless of

initialization. Similarly, for a shallower network, it would be easier to find a solution with higher rank embeddings.

2.4.4 Relation to random matrix theory

In linear models, we have a special case in which the low-rank embedding corresponds to low-rank weights. This enables us to make a natural connection to existing theoretical work from random matrix theory (RMT), which studies the spectral distribution under matrix multilpications (Akemann, Ipsen, and Kieburg 2013; Akemann, Kieburg, and Wei 2013; Burda, Jarosz, Livan, et al. 2010). We leverage the results from (Pennington, S. S. Schoenholz, and Ganguli 2017; Neuschel 2014) to show the following:

Theorem 2.4.1. *Let ρ be the effective rank measure defined in Definition 2.3.1. For a linear neural network with d-layers, where the parameters are drawn from the same Normal distribution $\{W_i\}_{i=1}^{d} \sim \mathcal{W}$, the effective rank of the weights monotonically decreases when increasing the number of layers when $dim(W) \to \infty$,*

$$\rho\left(W_d W_{d-1} \dots W_1\right) \leq \rho\left(W_{d-1} \dots W_1\right)$$

Proof. See Appendix 7.7.

Given the current set of mathematical tools, our preliminary theory depends on many assumptions, such as infinite width networks and the distribution of the weights; this is akin to many existing theoretical works. Yet, we have observed in practice that the empirical spectral distribution of finite-width models is well approximated by random matrix theory (see Appendix 7.1) in practice. We emphasize that the main contribution of our work is on the empirical theory of the low-rank bias of deep networks; nonetheless, we show that there is a natural theoretical connection to RMT in hopes of stimulating future works.

2.5 Over-parameterization as a regularizer

Thus far, we have observed that depth acts as a bias for finding functions with low-rank embeddings. As one could imagine, this inductive bias of depth could be used to help but also hurt generalization performance. Our observations indicate that the low-rank simplicity bias helps when the true function we are trying to approximate is low-rank. On the contrary, if the underlying mapping is a high rank or the network is made too deep, depth could have a converse effect on generalization. Ample evidence from prior works (Szegedy, Liu, Jia, et al. 2015; He, Zhang, Ren, et al. 2016) suggests that over-parameterization of non-linear models improves generalization on fixed datasets, but blindly increasing the number of layers without bells & whistles (e.g., batch-norm, residual connection, etc.) hurts (He, Zhang, Ren, et al. 2016).

Fortunately, networks are trained on natural data, where often the goal is to discover a low-rank relationship between the input and the label. Hence, the inductive bias of depth acts as a prior rather than a bug. As noted by (Solomonoff 1964b) theory of inductive inference, the simplest solution is often the best solution, suggesting that low-rank mapping in neural

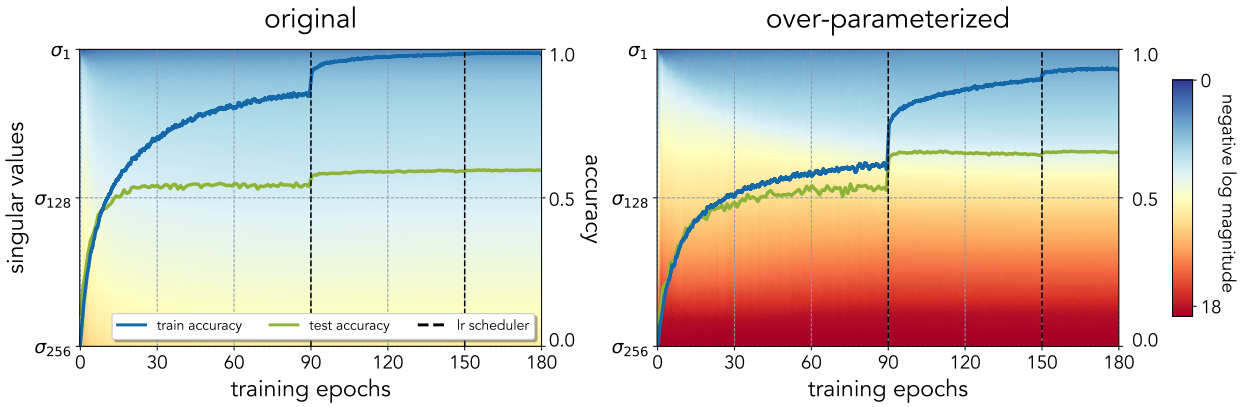


Figure 2.7: **Training dynamics:** Singular values of the Gram matrix for both original (left) and linearly over-parameterized (right) model throughout training. The models are trained on `CIFAR100` using SGD. Since the first few singular values dominate the distribution, we plot the negative log magnitude of the normalized singular values to better visualize how the intermediate singular values change. The singular values are sorted from largest to smallest $\sigma_i < \sigma_{i+1}$ (top to bottom in the figure) where blue means large and red means small. The original and the over-parameterized models are functionally equivalent and use the same colorbar and scale. The dotted lines (−−) indicate the learning step schedule, and train and test accuracies are overlayed on top of the distribution. The over-parameterized model learns lower rank embedding and exhibits less overfitting, and has better generalization. See Figure 7.7 and Figure 7.8 in the Appendix for the dynamics of the individual weights.

networks can be used to improve generalization and robustness to overfitting. However, increasing the number of non-linear layers also increases the modeling capacity, thereby making it difficult to isolate the effect of depth.

Nevertheless, since a non-linear network is composed of many linear components, such as fully connected and convolutional layers, we can over-parameterize these linear layers to induce a low-rank bias in the model without increasing the modeling capacity. The details of our linear over-parameterization method are in Appendix 7.2. We observe that such linear over-parameterization improves generalization performance on classification tasks. Furthermore, we find that such implicit regularization outperforms models trained with several choices of explicit regularization. (Guo, Alvarez, and Salzmann 2020) made a similar empirical observation in the context of model compression where linear over-parameterization improves generalization, but why it works is unexplored.

2.5.1 Image classification with over-parameterization

Using the linear expansion rules in Appendix 7.2, we over-parameterize various architectures and evaluate on a suite of standard image classification datasets: `CIFAR10`, `CIFAR100`, `ImageNet`. All models are trained using SGD with a momentum of 0.9. For data augmentation, we apply a random horizontal flip and random-resized crop. We follow standard training procedures and only modify the network architecture (see Appendix 7.5).

Expansion			CIFAR10		CIFAR100	
Factor	FC	Conv	accuracy	gain ↑	accuracy	gain ↑
1x	-	-	86.9	-	57.0	-
2x	✓	-	87.1	+0.2	58.4	+1.4
2x	-	✓	87.8	+0.9	61.0	+4.0
2x	✓	✓	**89.1**	**+2.2**	61.2	+4.2
4x	✓	-	87.3	+0.4	59.7	+2.7
4x	-	✓	**89.1**	**+2.2**	61.3	+4.3
4x	✓	✓	89.0	+2.1	**63.5**	**+6.5**
8x	✓	-	85.9	-1.0	58.8	+1.8
8x	-	✓	88.5	+1.6	61.6	+4.6
8x	✓	✓	88.0	+1.1	61.5	+4.5

Table 2.1: **Over-parameterization ablations:** A nonlinear CNN with 4 convolution and 2 linear layers trained on `CIFAR10` and `CIFAR100` with various degrees of linear over-parameterization.

architecture	ImageNet		
	original	over-param	gain ↑
AlexNet (2x)	57.3	59.1	**+1.8**
ResNet10 (2x)	62.8	63.7	**+0.9**
ResNet18 (2x)	67.3	67.7	**+0.4**

Table 2.2: **ImageNet:** We show on existing architectures that linear over-parameterization can improve generalization performance. The benefit plateaus off when using deeper models. We did not see a noticeable improvement starting from `ResNet34`.

In Figure 2.7, we compare a CNN trained without (left) and with (right) our over-parameterization (expansion factor $d = 4$) on `CIFAR100`. The CNN consists of 4 convolutional layers and 2 fully connected layers; the architecture details are in Appendix 7.5. We overlay the dynamics of the singular values of the Gram matrix throughout training. The spectral distribution is normalized by the largest singular value and are sorted in descending order $\sigma_i(A) \geq \sigma_{i+1}(A)$ for $i < 1 \leq \min(m, n)$. We observe that both the individual effective weights and the Gram matrix of the over-parameterized model is biased towards low-rank weights. Unlike the original, the majority of the singular values of the over-parameterized model are close to zero. When we take a closer look at the weights of the model, both the original and linearly over-parameterized models first exhibit rank contracting behavior throughout training, and then the rank starts to increase again – to the best of our knowledge, this is an unexpected training behavior in larger models that are not explained in prior works, possibly because the isometric, balanced initialization, and infinitesimal assumptions made in prior theoretical works do not hold in practice (visualized in Appendix 7.4).

We further quantify the gain in performance from linear over-parameterization in Table 2.1. The learning rate is tuned per configuration, and we report the best test accuracy throughout the training. We try various over-parameterization configurations and find an expansion factor of 4 to be the sweet spot, with a gain of +6.3 for `CIFAR100` and +2.8 for `CIFAR10`. The optimal expansion factor depends on the depth of the original network, and in general,

regularization	CIFAR10		CIFAR100	
	accuracy	gain ↑	accuracy	gain ↑
none (baseline)	86.9	-	57.0	-
low-rank initialization	86.8	-0.1	57.2	+0.2
ℓ_2 norm	87.2	+0.3	57.0	+0.0
ℓ_1 norm	87.4	+0.5	60.0	+3.0
nuclear norm	87.0	+0.1	58.1	+1.1
effective rank	86.9	+0.0	57.2	+0.2
stable rank	87.6	+0.9	58.3	+1.3
frobenius^2 norm	87.0	+0.1	59.2	+2.2
over-param (2x)	89.1	+2.2	61.2	+4.2
over-param (2x) + ℓ_2	89.6	+2.7	61.1	+4.1
over-param (2x) + ℓ_1	**89.7**	**+2.8**	**63.3**	**+6.3**

Table 2.3: **Explicit regularizers:** Comparison of models trained with various regularizers. Over-parameterization consistently outperforms explicit regularizers.

we observe a consistent improvement for over-parameterizing models with < 20 layers on image classification.

We scale up our experiments to ImageNet, a large-scale dataset consisting of 1.3 million images with 1000 classes, and show that our findings hold in practical settings. For these experiments, we use standardized architectures: AlexNet (Krizhevsky, Sutskever, and Hinton 2012) which consists of 8-layers, and ResNet10 / ResNet18 (He, Zhang, Ren, et al. 2016) which consists of 10 and 18 layers, respectively. If our prior observations hold true, we would expect the gain in performance from over-parameterization to be reduced for deeper models. This is, in fact, what we observed in Table 2.2, with moderate gains in AlexNet and less for ResNet10 and even less for ResNet18. In fact, starting from ResNet34, we observe linearly over-parameterized models perform worse than the original. These experiments support our claim that adding too many layers can over-penalize the model.

To find out whether explicit regularizers can approximate the advantages of over-parameterization, we directly compare the performance in Table 2.3 on `CIFAR`. These regularizers include popular ℓ_1 and ℓ_2 norm-based regularizers and commonly-used pseudo-measures of rank. These pseudo-measures of rank, such as *effective rank* and *nuclear norm*, require one to compute the singular value decomposition, which is computationally infeasible on large-scale models. Although we found explicit rank regularizers to help, we observed over-parameterization to outperform models trained with explicit regularizers. Moreover, we found that combining norm-based regularizers with over-parameterization further improves performance.

This discrepancy between implicit and explicit regularization may stem from the fact that over-parameterization receives a combined effect of both gradient descent's implicit bias and model parameterization's inductive bias. Therefore, one may need to jointly consider both biases to approximate its effect as an explicit regularizer correctly. Another reason could be that regularizers are inherently different than over-parameterization (Arora, Cohen, and Hazan 2018). For example, a model trained with a regularizer will have a non-zero gradient, even at zero training loss, while the over-parameterized model will not.

2.6 Discussion

One of the main ingredients in any machine learning algorithm is the choice of hypothesis space: what is the set of functions under consideration for fitting the data? Although this is a critical choice, *how the hypothesis space is also parameterized matters*. Even if two models span the same hypothesis space, the way we parameterize the hypothesis space can ultimately determine which solution the model will converge to – recent work has shown that networks with better neural reparameterizations can find more effective solutions (Hoyer, Sohl-Dickstein, and Greydanus 2019). The automation of finding the right parameterization also has a relationship to neural architecture search (Zoph and Le 2017), but architecture search typically conflates the search for better hypothesis spaces with the search for better parameterizations of a given hypothesis space. In this work, we have explored just one way of reparameterizing neural nets – stacking linear layers – which does not change the hypothesis space, but many other options exist (see Figure 7.3 and a short extension to residual networks Appendix 7.8). Understanding the biases induced by these reparameterizations may yield benefits in model analysis and design.

Chapter 3

The platonic representation hypothesis

In the previous chapter, we examined how deep neural networks have an inductive bias that favors simple solutions — deeper models are more likely to uncover simpler relationships that generalize better. If these models are indeed inclined toward simplicity, the number of ways to explain the data parsimoniously must then become increasingly limited.

This potential limitation could explain the recent convergence of AI systems, where models that perform well can be more easily integrated with each other. For instance, MLPs of LLMs can be seamlessly routed together, the weights of foundation models can be averaged with minimal interference (Wortsman, Ilharco, Gadre, et al. 2022), and vision and language AI systems can be aligned using a single MLP layer to create a multi-modal system (Liu, Li, Wu, et al. 2023). All this evidence suggests that their internal representations are strikingly similar (Bansal, Nakkiran, and Barak 2021). This convergence hints at a deeper truth: perhaps there is only *"one"* optimal way to be correct.

Evidence supporting this idea can be seen in the similarity of kernels across high-performing models. Despite differences in architecture or training data, these models tend to develop analogous internal structures. This phenomenon shouldn't be entirely surprising—just as the optimal program for solving Dijkstra's algorithm is likely to be similar across different programming languages.

This chapter builds on the previous chapter by positing that not only do deeper models prefer simple solutions, but they also tend to prefer the *same* simple solutions. We explain why and when this happens and provide a detailed discussion of the limitations of our hypothesis.

3.1 Introduction

AI systems are rapidly evolving into highly multifunctional entities. For example, whereas in the past we had special-purpose solutions for different language processing tasks (*e.g.*, sentiment analysis, parsing, dialogue), modern large language models (LLMs) are competent at all these tasks using a single set of weights (Srivastava, Rastogi, Rao, et al. 2022). Unified systems are also being built across data modalities: instead of using a different

architecture for processing images versus text, recent models, such as GPT4-V (OpenAI 2023), Gemini (Google 2023), and LLaVA (Liu, Li, Wu, et al. 2023), handle both modalities with a combined architecture.

More and more systems are built off of general-purpose pretrained backbones, sometimes called foundation models (Bommasani, Hudson, Adeli, et al. 2021), that support a large range of tasks, including robotics (Driess, Xia, Sajjadi, et al. 2023; Brohan, Brown, Carbajal, et al. 2023), bioinformatics (Ma, He, Li, et al. 2024), and healthcare (Steinberg, Jung, Fries, et al. 2021). In short, AI systems are becoming increasingly homogeneous in both their architectures and their capabilities.

This chapter explores one aspect of this trend: representational convergence. We argue that there is a growing similarity in how datapoints are represented in different neural network models. This similarity spans across different model architectures, training objectives, and even data modalities.

Our central hypothesis, stated above in Figure 3.1, is that there is indeed an endpoint to this convergence and a principle that drives it: different models are all trying to arrive at a *representation of reality*, meaning a representation of the joint distribution over events in the world that generate the data we observe. Figure 3.1 conveys this hypothesis: there exists a real world (labeled Z), which we measure with various sensors, such as the camera shown to the left (X). Other *projections* of these measurements, such as the textual description shown, can be produced from the first set of measurements or mediated by some other set of measurements, *e.g.*, touch or other camera views (dotted arrow from X to Y)[1].

Representation learning algorithms find vector embeddings that statistically model the various measurements and projections. The resulting vector embeddings are all derived from the underlying reality in Z and thereby become aligned. As models are trained on more data and for more tasks, they require representations that capture more and more information about Z, and hence alignment toward Z increases toward a convergent point as a function of scale.

We call this converged hypothetical representation the "platonic representation" in reference to Plato's Allegory of the Cave (Plato –0375), and his idea of an ideal reality that underlies our sensations. The training data for our algorithms are shadows on the cave wall, yet, we hypothesize, models are recovering ever better representations of the actual world outside the cave. This idea is not unique to Plato; our hypothesis is also related to the notion of "convergent realism" (Newton-Smith 1981; Putnam 1982; Doppelt 2007; Hardin and Rosenberg 1982) in the philosophy of science (*i.e.*, that science is converging on truth), and to many arguments that have been put forth in the representation learning literature (*e.g.*, Tian, Krishnan, and Isola 2020; Zimmermann, Sharma, Schneider, et al. 2021; Richens and Everitt 2024; Cao and Yamins 2021).

[1]Touch could convey the shapes in this example but not the colors. This is an important limitation to our hypothesis that we discuss at several points in the paper: different sensors and views might capture different information, which may limit their potential to converge to identical representations.

The Platonic Representation Hypothesis

Neural networks, trained with different objectives on different data and modalities, are converging to a shared statistical model of reality in their representation spaces.

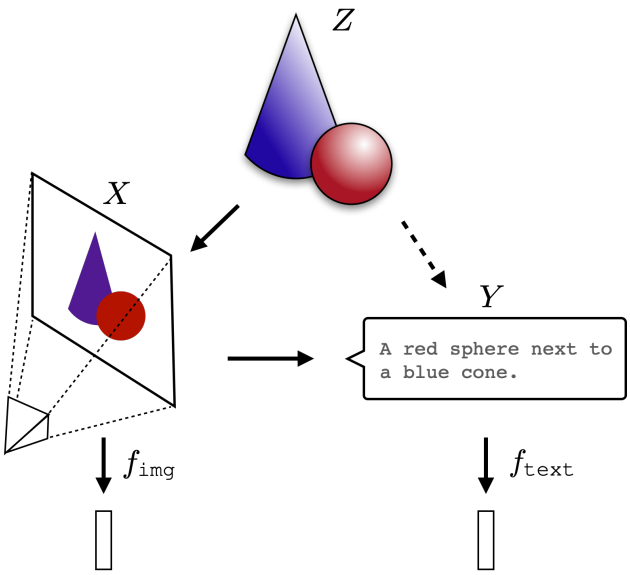


Figure 3.1: **The Platonic Representation Hypothesis:** Images (X) and text (Y) are projections of a common underlying reality (Z). We conjecture that representation learning algorithms will converge on a shared representation of Z, and scaling model size, as well as data and task diversity, drives this convergence.

3.2 Representations are converging

3.2.1 Preliminaries

We restrict our attention to representations that are *vector embeddings*. We characterize such a representation by the similarity structure it induces, referred to as its kernel. Kernels are commonly used to assess representations (Kornblith, Norouzi, Lee, et al. 2019; Klabunde, Schumacher, Strohmaier, et al. 2023); this can be justified by the fact that they capture the relative structures among data samples, which are also the learning signal for many machine learning algorithms (Aronszajn 1950; Smola and Schölkopf 1998). Following prior literature, we define *representational alignment* as a measure of the similarity of the similarity structures induced by two representations, *i.e.*, a similarity metric over kernels.

We give the mathematical definition of these concepts below:

- A **representation** is a function $f\colon \mathcal{X} \to \mathbb{R}^n$ that assigns a feature vector to each input in some data domain $\mathcal{X}$.
- A **kernel**, $K\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, characterizes how a representation measures distance/similarity between datapoints. $K(x_i, x_j) = \langle f(x_i), f(x_j) \rangle$, where $\langle \cdot, \cdot \rangle$ denotes inner product,

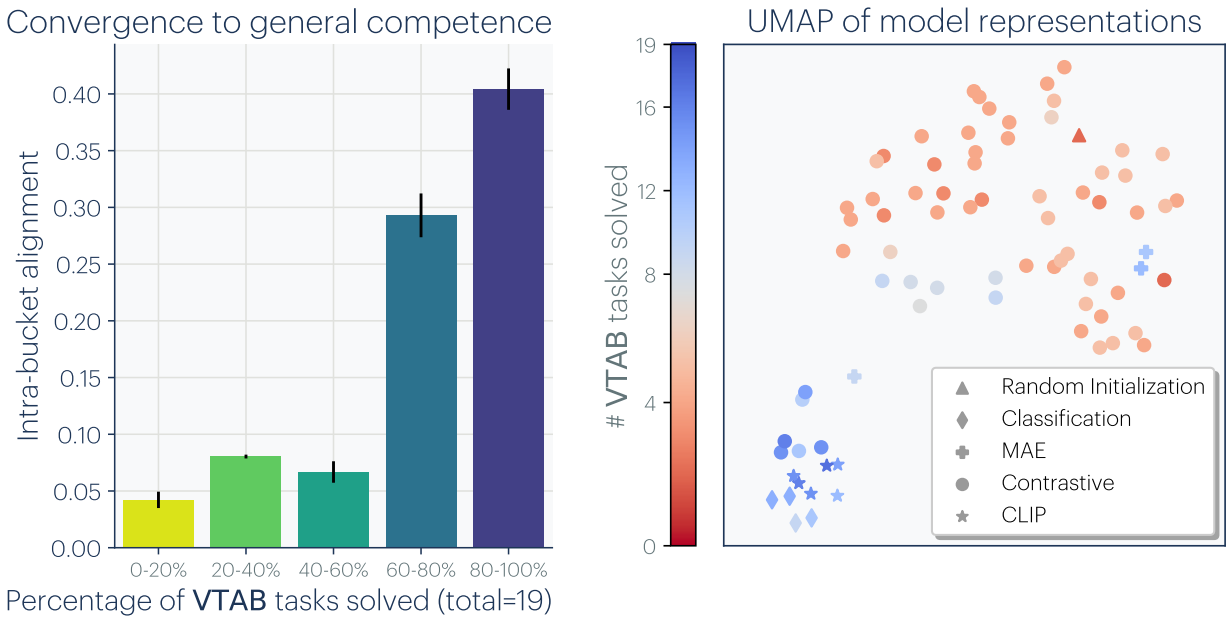


Figure 3.2: **VISION models converge as COMPETENCE increases:** We measure alignment among 78 models using mutual nearest-neighbors on Places-365 (Zhou, Lapedriza, Khosla, et al. 2017), and evaluate their performance on downstream tasks from the Visual Task Adaptation Benchmark (VTAB; (Zhai, Puigcerver, Kolesnikov, et al. 2019)). **LEFT:** Models that solve more VTAB tasks tend to be more aligned with each other. Error bars show standard error. **RIGHT:** We use UMAP to embed *models* into a 2D space, based on $\mathsf{distance} \triangleq -\log(\mathsf{alignment})$. More competent and general models (blue) have more similar representations.

$x_i, x_j \in \mathcal{X}$ and $K \in \mathcal{K}$.

- A **kernel-alignment metric**, $m\colon \mathcal{K} \times \mathcal{K} \to \mathbb{R}$, measures the similarity between two kernels, *i.e.*, how similar is the distance measure induced by one representation to the distance measure induced by another. Examples include Centered Kernel Distance (CKA) (Kornblith, Norouzi, Lee, et al. 2019), SVCCA (Raghu, Gilmer, Yosinski, et al. 2017), and nearest-neighbor metrics (Klabunde, Schumacher, Strohmaier, et al. 2023).

In our experiments, we use a *mutual nearest-neighbor metric* that measures the mean intersection of the k-nearest neighbor sets induced by two kernels, K_1 and K_2, normalized by k. This metric is a variant of those proposed in Park, Wang, Ardeshir, et al. (2024), Klabunde, Schumacher, Strohmaier, et al. (2023) and Oron, Dekel, Xue, et al. (2017). See Section 6.1 for the exact definition and Section 6.2 for comparisons with alternative alignment metrics.

Next, we explore several ways in which representations are converging. First, we argue that different neural networks are converging to aligned representations. Then, we show that this continues to hold across modalities, where image embeddings in vision models align with text embeddings in language models.

3.2.2 Different models, with different architectures and objectives, can have aligned representations

One indication of representational convergence is the rising number of systems built on top of pre-trained foundation models. These models are becoming standard backbones across a growing spectrum of tasks. Their versatility across numerous applications implies a level of universality in the way they represent data.

While this trend implies convergence toward a relatively small set of foundation models, it does not imply that *different* foundation models will arrive at the same representation. Yet that is what has been observed by several recent papers.

Lenc and Vedaldi (2015) conducted one such study, in which they measured representational similarity through a technique called *model stitching*. Given two models, f and g, each composed of multiple layers ($f = f_1 \circ \cdots \circ f_n$, $g = g_1 \circ \cdots \circ g_m$), an intermediate representation from f is integrated into g via a learned affine stitching layer h, resulting in a new stitched model $F = f_1 \circ \cdots \circ f_k \circ h \circ g_{k+1} \circ \cdots \circ g_m$. If F has good performance, it indicates that f and g have compatible representations at layer k, up to the transform h.

In their study, Lenc and Vedaldi (2015) made two notable findings: (1) A vision model trained on ImageNet (Russakovsky, Deng, Su, et al. 2015) can be aligned with a model trained on Places-365 (Zhou, Lapedriza, Khosla, et al. 2017) while maintaining good performance; (2) The early layers of these convolutional networks are more interchangeable than later layers. The first finding illustrates a level of data independence where distinct image datasets lead to similar representations. The second finding agrees with extensive research that oriented Gabor-like filters are common in both artificial and biological vision systems. This suggests a convergence to a similar initial layer of representation across various neural network architectures (Olshausen and Field 1996; Krizhevsky, Sutskever, and Hinton 2017). Bansal, Nakkiran, and Barak (2021) expanded on the idea of model stitching, showing that models trained using self-supervised objectives align closely with their supervised counterparts.

Moschella, Maiorca, Fumero, et al. (2022) further demonstrated the feasibility of "zero-shot" model stitching without learning a stitching layer. Despite the fact that different text models were trained on different modalities, they found that the models often embed data in remarkably similar ways. In particular, they considered the kernel K defined by learned representations and showed that K serves as a bridge between models, allowing an encoder trained in one language, like English, to work effectively with a decoder in another, like French.

Dravid, Gandelsman, Efros, et al. (2023) extended this idea to individual neurons, and found "Rosetta Neurons" that are activated by the same pattern across a range of vision models. Such neurons form a common dictionary independently discovered by all models.

3.2.3 Alignment increases with scale and performance

Kornblith, Norouzi, Lee, et al. (2019) and Roeder, Metz, and Kingma (2021) observed model alignment not only exists but also increases with model scale and dataset size. On CIFAR-10

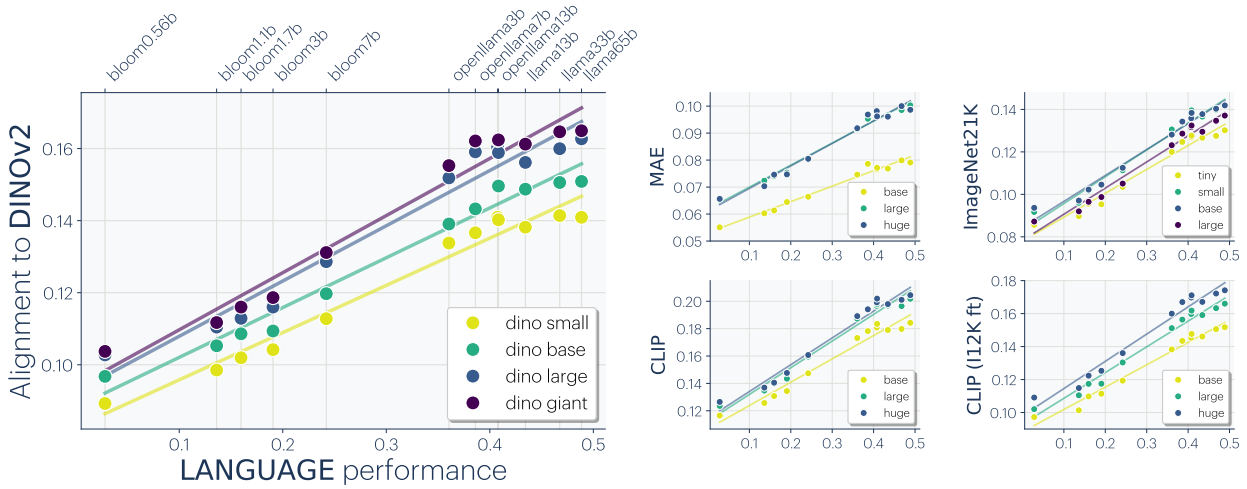


Figure 3.3: **LANGUAGE and VISION models align:** We measure alignment using mutual nearest-neighbor on the Wikipedia caption dataset (WIT) (Srinivasan, Raman, Chen, et al. 2021). The x-axis is the language model performance measured over 4M tokens from the OpenWebText dataset (Gokaslan and Cohen 2019) (see Appendix 6.2 for plots with model names). We measure performance using $1 - \texttt{bits-per-byte}$, where `bits-per-byte` normalizes the cross-entropy by the total bytes in the input text string. The results show a linear relationship between language-vision alignment and language modeling score, where a general trend is that more capable language models align better with more capable vision models. We find that CLIP models, which are trained with explicit language supervision, exhibit a higher level of alignment. However, this alignment decreases after being fine-tuned on ImageNet classification (labeled CLIP (I12K ft)).

classification, Krizhevsky, Hinton, et al. (2009) found that larger models exhibit greater alignment with each other compared to smaller ones. Theoretically, Balestriero and Baraniuk (2018) showed that models with similar outputs (*e.g.*, as a result of having high performance) also have similar internal activations. With the continuing trend of models scaling up, this suggests model alignment will increase over time – we might expect that the next generation of bigger, better models will be even more aligned with each other.

We expand upon this observation by evaluating the transfer performance of 78 vision models. These models were trained with varying architectures, training objectives, and datasets (detailed in Section 6.3.1). In Figure 3.2 (left), we bin these models based on their average transfer performance on the VTAB dataset (Zhai, Puigcerver, Kolesnikov, et al. 2019), and then measure the average kernel alignment of the models within each bin. The results indicate that models with high transfer performance form a tightly clustered set of representations, while models with weak performance have more variable representations. We further visualize this structure with UMAP (McInnes, Healy, and Melville 2018) over models representation in Figure 3.2 (right). This suggests that models that are competent all represent data in a similar way. Echoing Bansal, Nakkiran, and Barak (2021) and Tolstoy (1877), we might say: all strong models are alike, each weak model is weak in its own way.

The discussion so far indicates that various models are aligning toward a unified representation. But does the convergence extend to model weights? While models with different architectures

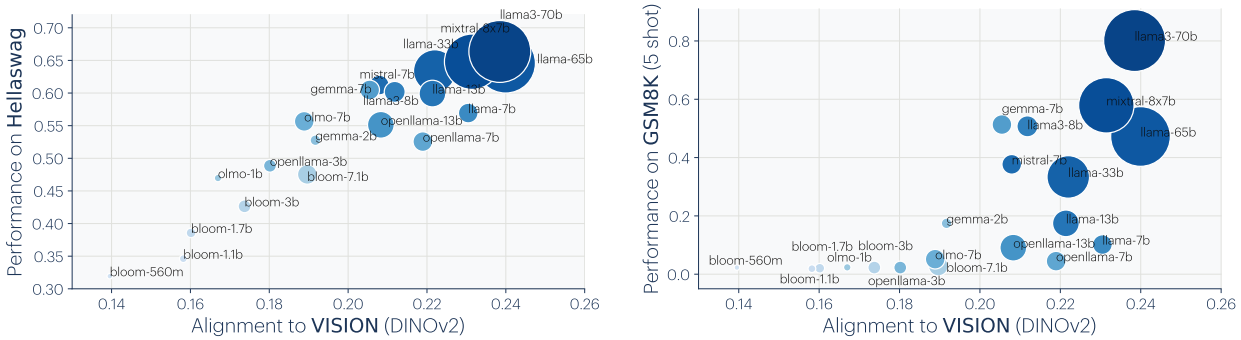


Figure 3.4: **Alignment predicts downstream performance:** We visualize correlation between LLM alignment score to DINOv2 (Oquab, Darcet, Moutakanni, et al. 2023) and downstream task performance on Hellaswag (common-sense) (Zellers, Holtzman, Bisk, et al. 2019) and GSM8K (math) (Cobbe, Kosaraju, Bavarian, et al. 2021). LLMs are plotted with radii proportional to the size of the model, and color-coded by their rank order in language modeling scores ($1 - \texttt{bits-per-byte}$). We observe that models aligned more closely with vision also show better performance on downstream language tasks. For Hellaswag, there is a linear relationship with alignment score, while GSM8K exhibits an "emergence"-esque trend.

might not have compatible weight spaces, there exists ample evidence that models with the same architecture will often converge to the same basin of weights (Nagarajan and Kolter 2019; Garipov, Izmailov, Podoprikhin, et al. 2018; Lubana, Bigelow, Dick, et al. 2023). This holds even for models with different initializations, up to permutations over weight space (S. K. Ainsworth, Hayase, and Srinivasa 2022). Because of this, it is possible to merge separately trained models of the same architecture, and achieve some of the capabilities of all models in the mixture (Stoica, Bolya, Bjorner, et al. 2023; Jordan, Sedghi, Saukh, et al. 2022; Wortsman, Ilharco, Gadre, et al. 2022).

3.2.4 Representations are converging across modalities

Do models trained on different data modalities also converge? Several works indicate that the answer is *yes*.

Merullo, Castricato, Eickhoff, et al. (2022) extended model stitching to the cross-modal setting, finding that a single linear projection is sufficient to stitch a vision model to an LLM and achieve good performance on visual question answering and image captioning. Koh, Salakhutdinov, and Fried (2023) showed that linear stitching can also work in the opposite direction, aligning text inputs to visual outputs. In fact, many recent language-vision models stitch pre-trained language and vision models together. For example, LLaVA (Liu, Li, Wu, et al. 2023) demonstrated state-of-the-art results by projecting visual features into a language model with a 2-layer MLP.

Other works show further kinds of evidence of cross-modal synergy. OpenAI (2023) found that jointly training a language model with a vision model improves performance on language tasks, compared to training the language model on its own. Sorscher, Ganguli, and Sompolinsky (2022) show a setting in which word embeddings of visual concept names can

be isometrically mapped to image embeddings for those same concepts. In work concurrent to ours, Maniparambil, Akshulakov, Djilali, et al. (2024) show well-trained vision encoders on large datasets exhibit high semantic similarity with language encoders regardless of the training paradigm (supervised, self-supervised, or language-supervised). Sharma, Rott Shaham, Baradad, et al. (2024) probed the visual knowledge of LLMs trained *only* on language data, by converting images into code that an LLM can process. They found that LLMs have rich knowledge of visual structures, to the extent that decent visual representations can be trained on images generated solely by querying an LLM to produce code and rendering the response. In visual generation, LLMs show abilities to augment captions with visual structures (*e.g.*, bounding boxes) and improve generation quality (Betker, Goh, Jing, et al. 2023; Lian, Li, Yala, et al. 2023; Lian, Shi, Yala, et al. 2023; Wu, Lian, Gonzalez, et al. 2023). Over other modalities, Ngo and Kim (2024) showed auditory models are also roughly aligned with LLMs up to a linear transformation, and Ng, Subramanian, Klein, et al. (2023) demonstrated the effectiveness of using pre-trained LLMs for facial motion prediction.

We set out to address these claims in a broader scope to determine whether models are indeed learning an increasingly modality-agnostic representation of the world. We sampled a variety of models trained either solely on vision or solely on language, and compared their representations as they became larger and more competent over many tasks.

In Figure 3.3, we assess alignment between a suite of language models and vision models. So far we have only defined alignment for two kernels defined over the same input space. To measure cross-modal alignment, we use paired datasets to bridge the two modalities. For vision and text, we use the Wikipedia captions dataset $\{(x_i, y_i)\}_i$ (Srinivasan, Raman, Chen, et al. 2021), composed of images from Wikipedia (x_i) and their corresponding captions (y_i). We then measure alignment between a language model $f_{\texttt{text}}$ and a vision model $f_{\texttt{img}}$ as the alignment of the two following kernels:

$$K_{\texttt{img}}(i, j) = \langle f_{\texttt{img}}(x_i), f_{\texttt{img}}(x_j) \rangle \tag{3.1}$$

$$K_{\texttt{text}}(i, j) = \langle f_{\texttt{text}}(y_i), f_{\texttt{text}}(y_j) \rangle. \tag{3.2}$$

Using this analysis, we find that the better an LLM is at language modeling, the more it tends to aligns with vision models, as shown in Figure 3.3. The converse effect also holds: the better a vision models is, the more it tends to align with LLMs. See Section 6.3.2 for more details.

3.2.5 Models are increasingly aligning to brains

Neural networks also show substantial alignment with biological representations in the brain (Yamins, Hong, Cadieu, et al. 2014). This commonality may be due to similarities in the task and data constraints both systems are confronted with.

Even though the mediums may differ – silicon transistors versus biological neurons – the fundamental problem faced by brains and machines is the same: efficiently extracting and understanding the underlying structure in images, text, sounds, *etc.* (Barlow et al. 1961; Olshausen and Field 1997). Sorscher, Ganguli, and Sompolinsky (2022) developed a theoretical

framework for how the efficient extraction of novel concepts occurs for both the human visual system and deep networks. The tasks that the human visual system has been honed to perform through evolution – like segmentation, detection, and whole-image classification – are also the ones that we train our neural nets to perform. Yamins, Hong, Cadieu, et al. (2014) went as far as to title their work in the spirit that performance over such tasks implies brain alignment. Antonello and Huth (2024) posited that it is less the particular task and more the generality of the representations that explain their alignment with biological representations. Further, Conwell, Prince, Kay, et al. (2022) showed that training data plays a large role in alignment. Psychophysical studies have also shown agreement between how humans perceive visual similarity and how models do, even when the models are trained on tasks, such as self-supervised prediction, that are seemingly unrelated to mimicking human perception (Zhang, Isola, Efros, et al. 2018).

3.2.6 Does alignment predict downstream performance?

If models are converging towards a more accurate representation of reality, we expect that alignment should correspond to improved performance on downstream tasks. Figure 3.4 supports this hypothesis by demonstrating improved performance on commonsense reasoning (Hellaswag; Zellers, Holtzman, Bisk, et al. (2019)) and mathematical problem solving (GSM8K; Cobbe, Kosaraju, Bavarian, et al. (2021)) as alignment improves.

3.3 Why are representations converging?

Modern machine learning models are typically trained to minimize empirical risk, often with implicit and/or explicit regularization. This process can be formally expressed as:

$$\overbrace{f^*}^{\text{trained model}} = \arg\min_{f \in \underbrace{\mathcal{F}}_{\text{function class}}} \mathbb{E}_{x \sim \text{dataset}} [\overbrace{\mathcal{L}}^{\text{training objective}}(f, x)] + \underbrace{\mathcal{R}}_{\text{regularization}}(f)$$

By dissecting these elements, we aim to better understand the mechanism that drives model convergence.

3.3.1 Convergence via Task Generality

Each training datapoint and objective (task) places an additional constraint on the model. As data and tasks scale, the volume of representations that satisfy these constraints must proportionately grow smaller, as visualized in Figure 3.6 and stated below:

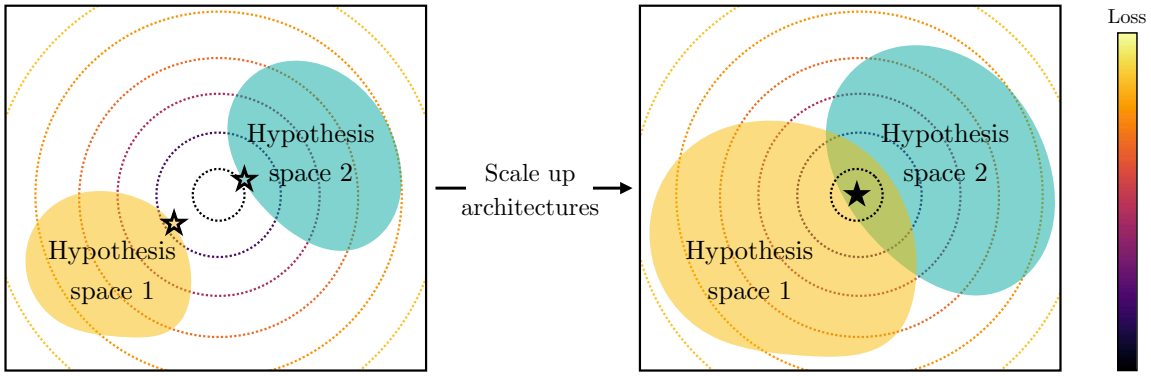


Figure 3.5: **The Capacity Hypothesis:** If an optimal representation exists in function space, larger hypothesis spaces are more likely to cover it. **LEFT:** Two small models might not cover the optimum and thus find *different* solutions (marked by outlined ☆). **RIGHT:** As the models become larger, they cover the optimum and converge to the same solution (marked by filled ★).

The Multitask Scaling Hypothesis

There are fewer representations that are competent for N tasks than there are for $M < N$ tasks. As we train more general models that solve more tasks at once, we should expect fewer possible solutions.

This has been previously termed as the Contravariance principle by Cao and Yamins (2021), which states that the set of solutions to an easy goal is large, while the set of solutions to a challenging goal is comparatively smaller. Moreover, we argue that this narrower solution set also generalizes better. As data scales, models that optimize the empirical risk $\mathbb{E}_{x\sim\,\mathsf{dataset}}[\mathcal{L}(f,x)]$ also improve on the population risk $\mathbb{E}_{x\sim\,\mathsf{reality}}[\mathcal{L}(f,x)]$, and become better at capturing statistical structures of the true data generating process (reality).

Recent work has demonstrated a power law relationship between data scale and model performance (Hestness, Narang, Ardalani, et al. 2017). This implies that with enough data (*e.g.*, consisting of the entire internet and all offline scientific measurements) one ought to converge to a very small solution set with irreducible error – the inherent epistemic uncertainty of the world. As more models are trained on internet-scale data, the set of solutions that satisfies all data constraints must become relatively small.

In addition to data-scaling, many modern representation learning objectives $\mathcal{L}(f,x)$ directly optimize for multi-task solving. Contrastive learning finds a distance structure over data samples that optimizes many classification tasks (Arora, Khandeparkar, Khodak, et al. 2019; Wang and Isola 2020; Tian, Wang, Krishnan, et al. 2020). Masked Autoencoders (He, Chen, Xie, et al. 2021) optimize randomly sampled reconstruction tasks. In fact, autoregressive language modeling can also be seen as optimizing a diverse set of tasks (Radford, Wu, Child, et al. 2019). Such multi-task objectives may be more effective than single-task ones (*e.g.*, ImageNet classification) due to the fact that they impose more task constraints on the representation, leading to a smaller and higher-quality solution space (Chen, Kornblith,

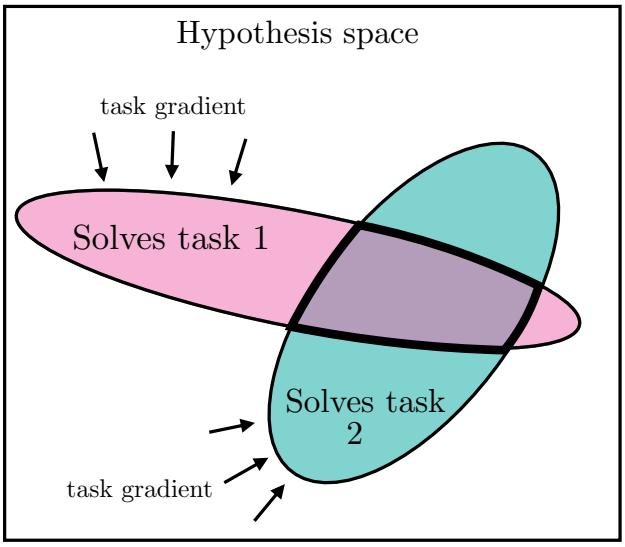


Figure 3.6: **The Multitask Scaling Hypothesis:** Models trained with an increasing number of tasks are subjected to pressure to learn a representation that can solve all the tasks.

Norouzi, et al. 2020; He, Fan, Wu, et al. 2020; Radford, Jozefowicz, and Sutskever 2017; Radford, Wu, Child, et al. 2019).

3.3.2 Convergence via Model Capacity

Suppose there is a globally optimal representation for standard learning objectives. Then, under sufficient data, *scaling* a model (*i.e.*, using larger function classes $\mathcal{F}$), as well as improved optimization , should be more effective at finding better approximations to this optimum, as illustrated in Figure 3.5.

With the same training objective, larger models, even of different architectures, will thus tend to converge toward this optimum. When different training objectives share similar minimizers, larger models are better at finding these minimizers, and will train to similar solutions over the training tasks. We summarize this hypothesis as follows:

The Capacity Hypothesis

Bigger models are more likely to converge to a shared representation than smaller models.

3.3.3 Convergence via Simplicity Bias

The forces driving representational convergence often appear to be in tension with one another. On one hand, convergence through increased task complexity and model capacity suggests a need to expand the model's hypothesis class. On the other hand, this expansion must be outpaced by an even faster growth in the available data. This balance is crucial: if the model's capacity grows as rapidly as the dataset, we might expect the model to remain

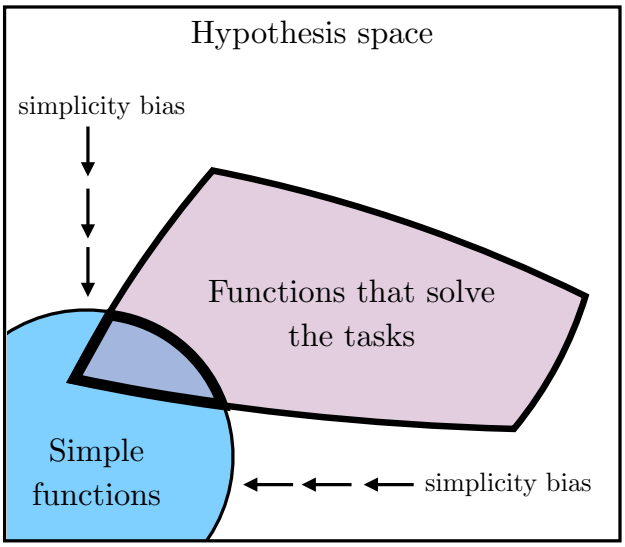


Figure 3.7: **The Simplicity Bias Hypothesis:** Larger models have larger coverage of all possible ways to fit the same data. However, the implicit simplicity biases of deep networks encourage larger models to find the simplest of these solutions.

equally capable of learning multiple, potentially distinct solutions.

Furthermore, in the over-parameterized regime – where model capacity significantly exceeds the minimum required to fit the training data – models may achieve identical performance on the training set while developing fundamentally different internal representations. These hypothesis alone cannot fully characterize why their internal representations would align. What would stop a billion-parameter (and counting) model from learning an overly complicated and distinct representation? One key factor might be the simplicity bias:

The Simplicity Bias Hypothesis

Deep networks are biased toward finding simple fits to the data, and the bigger the model, the stronger the bias. Therefore, as models get bigger, we should expect convergence to a smaller solution space.

Such simplicity bias could be coming from explicit regularization $\mathcal{R}(f)$ commonly used in deep learning (*e.g.*, weight decay and dropout). However, even in the absence of external influences, deep networks naturally adhere to Occam's razor, implicitly favoring simple solutions that fit the data (Solomonoff 1964b; Gunasekar, Lee, Soudry, et al. 2018; Arora, Cohen, Hu, et al. 2019; Valle-Perez, Camargo, and Louis 2019; Huh, Mobahi, Zhang, et al. 2023; Dingle, Camargo, and Louis 2018; Goldblum, Finzi, Rowan, et al. 2023). Figure 3.7 visualizes how simplicity bias can drive convergence.

3.4 What representation are we converging to?

By now, we hope to have convinced the reader that task and data pressures, combined with increasing model capacity, can lead to convergence. We next turn our attention to *what* exactly is the endpoint of all this convergence.

Our central hypothesis, stated in Figure 3.1, is that the representation we are converging toward is a statistical model of the underlying reality that generates our observations. Consistent with the multitask scaling hypothesis, such a representation would naturally be useful toward many tasks (or at least toward any task grounded in reality). Additionally, this representation might be relatively simple, assuming that scientists are correct in suggesting that the fundamental laws of nature are indeed simple functions (Gell-Mann 1995), in line with the simplicity bias hypothesis.

But what exactly do we mean by "a statistical model of the underlying reality." In this section, we formalize one definition with concrete mathematical statements. *Importantly*, this section should be read as just one concrete candidate for the form of the platonic representation; other candidates could be arrived at from other modeling assumptions.

3.4.1 An idealized world

We consider a world that works as follows, consistent with the cartoon in Figure Figure 3.1. The world consists of a sequence of T discrete events, denoted as $\mathbf{Z} \triangleq [z_1, \ldots, z_T]$, sampled from some unknown distribution $\mathbb{P}(\mathbf{Z})$. Each event can be observed in various ways. An observation is a bijective, deterministic function $\texttt{obs} : \mathcal{Z} \to \cdot$ that maps events to an arbitrary measurement space, such as pixels, sounds, mass, force, torque, words, etc. Later, in Section 3.6, we discuss limitations and potential extensions to continuous and unbounded worlds, and stochastic observations, that could yield a model that better reflects real learning scenarios.

One can think of an event as corresponding to the state of the world at some point in time[2], but it is also fine to simply consider an event as any variable that indexes observations, with no further physical meaning[3].

In this idealized world, knowing $\mathbb{P}(\mathbf{Z})$ would be useful for many kinds of predictions; this would constitute a world model over the events that cause our observations (Werbos 1987; Ha and Schmidhuber 2018; Richens and Everitt 2024). We will next show that a particular representation of $\mathbb{P}(\mathbf{Z})$ is recovered by certain contrastive learners.

[2]Here we only analyze temporal sequences, but note that the same could be done with respect to events laid out in space instead.

[3]This latter interpretation may be more consistent with Plato's intent. Scholars have argued that his allegory of the cave rejects any notion of a true world state (Nettleship 1897). Instead, we could say that the joint distribution of observation indices is *itself* the platonic reality.

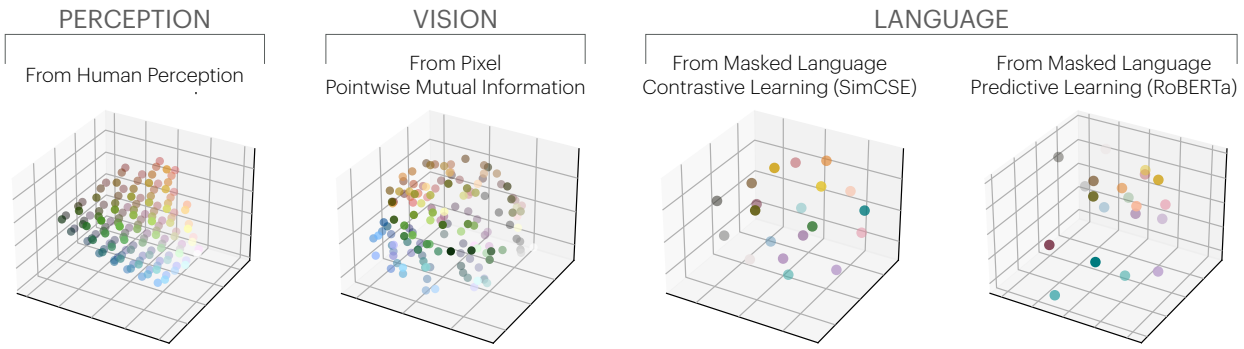


Figure 3.8: **Color cooccurrence in VISION and LANGUAGE yields perceptual organization:** Similar representations of color are obtained via, **from LEFT to RIGHT**, the perceptual layout from CIELAB color space, cooccurrence in CIFAR-10 images, and language cooccurrence modeling (Gao, Yao, and Chen (2021) and Liu, Ott, Goyal, et al. (2019); computed roughly following Abdou, Kulmizev, Hershcovich, et al. (2021)). Details in Section 6.4.

3.4.2 A family of contrastive learners converge to a representation of $\mathbb{P}(\mathbf{Z})$

Consider a contrastive learner that models observations that *cooccur* together. For simplicity, we ground our discussion with the following definition of the *cooccurrence probability*, P_{coor}, of two observations x_a and x_b both occurring within some window T_{window}:

$$P_{\mathsf{coor}}(x_a, x_b) \quad \propto \sum_{(t,t')\colon |t-t'| \leq T_{\mathsf{window}}} \mathbb{P}(X_t = x_a, X_{t'} = x_b).$$

Analogously, we can define P_{coor} for $\mathbf{Z}$ and other observation modalities. Note that P_{coor} is symmetric.

Consider *positive pairs* as two observations nearby in time (sampled from P_{coor}) and *negative pairs* as observations drawn from any point in time (sampled independently from the marginal). Our contrastive learner tries to classify if a pair is positive or negative by learning a representation $f_X\colon X \to \mathbb{R}^d$ such that the dot-product kernel approximates the log odds ratio up to some offset:

$$\langle f_X(x_a), f_X(x_b) \rangle \approx \log \frac{\mathbb{P}(\mathtt{pos} \mid x_a, x_b)}{\mathbb{P}(\mathtt{neg} \mid x_a, x_b)} + \tilde{c}_X(x_a) \tag{3.3}$$

$$= \log \frac{P_{\mathsf{coor}}(x_a \mid x_b)}{P_{\mathsf{coor}}(x_a)} + c_X(x_a) \tag{3.4}$$

$$= K_{\mathsf{PMI}}(x_a, x_b) + c_X(x_a), \tag{3.5}$$

where K_{PMI} is the pointwise mutual information (PMI) kernel, and $c_X(x_a)$ is constant in x_b. We note that this is a common setting for self-supervised contrastive learners with

NCE objectives (Gutmann and Hyvärinen 2010; Oord, Li, and Vinyals 2018), including SimCLR (Chen, Kornblith, Norouzi, et al. 2020) and SimCSE (Gao, Yao, and Chen 2021). (See Oord, Li, and Vinyals (2018) and Section 6.6.1 for detailed derivations.)

Under mild conditions that the world is smooth enough (see Section 6.6.2), a choice of f_X can exactly represent K_{PMI}:

$$\langle f_X(x_a), f_X(x_b) \rangle = K_{\mathsf{PMI}}(x_a, x_b) + c_X, \tag{3.6}$$

where we observed that $c_X(x_a)$ from Equation (3.5) must be a constant since both sides are symmetric.

Therefore, the contrastive learners we consider are minimized by a representation f_X whose kernel is K_{PMI} (up to a constant offset). With sufficient data and optimization, we will observe convergence to this point.

Thus we have convergence to a representation of the statistics of X, but what about Z? Recall that our idealized world consists of *bijective* observation functions, which, over discrete random variables, preserve probabilities. So we have:

$$\begin{aligned} P_{\mathsf{coor}}(x_a, x_b) &= P_{\mathsf{coor}}(z_a, z_b) \\ K_{\mathsf{PMI}}(x_a, x_b) &= K_{\mathsf{PMI}}(z_a, z_b), \end{aligned}$$

where we use P_{coor} and K_{PMI} in a modality-agnostic way to emphasize that different modalities share the same such quantities.

All these arguments hold not just for X but also for Y (or any other bijective, discrete modality), implying:

$$\begin{aligned} K_{\mathsf{PMI}}(z_a, z_b) &= \langle f_X(x_a), f_X(x_b) \rangle - c_X & (3.7) \\ &= \langle f_Y(y_a), f_Y(y_b) \rangle - c_Y. & (3.8) \end{aligned}$$

Therefore, for any modality in our idealized world, we observe representational convergence to the same kernel, which represents certain pairwise statistics of $\mathbb{P}(\mathbf{Z})$.

This analysis suggests that certain representation learning algorithms may boil down to a simple rule: *find an embedding in which similarity equals PMI*. We note that this idea is consistent with prior works that have used PMI as a similarity measure for clustering in vision and language (*e.g.*, Isola, Zoran, Krishnan, et al. (2014), Isola (2015), Isola, Zoran, Krishnan, et al. (2016), and Chambers and Jurafsky (2008)).

A study in color We conduct a case study to verify that convergence does happen on real data. Abdou, Kulmizev, Hershcovich, et al. (2021) discovered that color distances in learned language representations, when trained to predict cooccurrences in *text* (Devlin, Chang, Lee, et al. 2018), closely mirror human perception of these distances, which we reproduce in Figure 3.8 with both contrastive and predictive models. Interestingly, they noted an increasing similarity as models scale larger and become better at modeling *text* cooccurrences. In Figure 3.8, we also learn representations of color based on K_{PMI} from cooccurrences in *images*. Indeed, learning cooccurrence statistics in either domain recovers roughly the *same* perceptual representation. Details of this experiment are described in Section 6.4.

We believe that our simple model encapsulates essential aspects of complex real-world systems, and offers a path toward understanding the representation that models are converging to – a unified model that is proficient across various domains and modalities, grounded in the statistical properties of the underlying world. Section 3.6 further elaborates some limitations.

3.5 What are the implications of convergence?

Scaling is sufficient, but not necessarily efficient Our arguments are roughly in line with the claim that "scale is all you need" to reach high levels of intelligence. We have argued that as resources are scaled (# parameters, # datapoints, # flops), representations are converging, regardless of other modeling choices and even data modality. Does this mean that scale is all that matters? Not quite: different methods can scale with different levels of *efficiency* (Hestness, Narang, Ardalani, et al. 2017; Kaplan, McCandlish, Henighan, et al. 2020), and successful methods must still satisfy some general requirements (*e.g.*, be a consistent estimator, model pairwise statistics of $\mathbb{P}(\mathbf{Z})$).

Training data can be shared across modalities Suppose you have access to N images and M sentences, and want to learn the best representation. If there is indeed a modality-agnostic platonic representation, then the image data should help find it, and so should the language data. The implication is that if you want to train the best vision model, you should train not just on N images but also on M sentences. This is already becoming common practice (OpenAI 2023; Radford, Kim, Hallacy, et al. 2021). Many vision models are finetuned from pre-trained LLMs. The other direction is less common, but also is implied by our hypothesis: if you want to build the best LLM, *you should also train it on image data.* Indeed, OpenAI (2023) claim evidence that this is true, where training on images improved performance on text. In theory, there should be some conversion ratio: a pixel is worth a words for training LLMs, and a word is worth b pixels for training vision models.

Ease of translation and adaptation across modalities When two representations are aligned, transitioning from one to the other should be a simple function that's easily obtained. Our hypothesis could explain the phenomenon that conditional generation is easier than unconditional (Mirza and Osindero 2014; Liu, Wang, Bau, et al. 2020; Sauer, Schwarz, and Geiger 2022), as the data we condition on may have the same platonic structure as the data we are generating. In line with this, recent work has found that representation-conditioning

is even easier (Li, Katabi, and He 2023). Similarly, representational convergence could act as a bridge that lets us find mappings between domains even without paired data; this may underlie the success of unpaired translation in vision (Zhu, Park, Isola, et al. 2017; Shi, De Bortoli, Campbell, et al. 2024; Xie, Ho, and Zhang 2022) and language (Tran, Burda, and Sutskever 2017; Lample, Ott, Conneau, et al. 2018).

We emphasize that this doesn't mean that models trained on a single modality (*e.g.*, language) can immediately process raw data from another (*e.g.*, vision). What makes them adaptable to the new modalities is that they share a common modality-agnostic representation, and can readily process *representations* of new modalities. Furthermore, this implies that language models would achieve some notion of grounding in the visual domain even in the absence of cross-modal data[4].

The primary advantage of cross-modal data could then simply be sample efficiency.

Scaling may reduce hallucination and bias A prominent shortcoming of current LLMs is their propensity to hallucinate, or output false statements. If models are indeed converging toward an accurate model of reality, and scale powers this convergence, then we may expect hallucinations to decrease with scale. Of course, our hypothesis is conditioned on the training data for future models constituting a sufficiently lossless and diverse set of measurements. This may not come to pass, but it is an implication of our hypothesis worth pointing out. A similar argument can be made about certain kinds of bias. It has been shown that large models can exacerbate existing biases present in their training data (Hall, Maaten, Gustafson, et al. 2022). Our hypothesis implies that, while this may be true, we should expect *larger* models to amplify bias *less*. This does not mean bias will be removed, rather that the model's biases will more accurately reflect the data's biases, rather than exacerbating them.

3.6 Counterexamples and limitations

Different modalities may contain different information One immediate objection to our hypothesis is: what about the information that is unique to a given modality? Can language really describe the ineffable experience of watching a total solar eclipse? Or, how could an image convey the a concept like "I believe in the freedom of speech," which is easy to write in English? Two different models cannot converge to the same representation if they have access to fundamentally different information.

More precisely, our mathematical argument in Section 3.4 only strictly holds for bijective projections of $\mathbf{Z}$, so that the information in all the projections is equivalent to the information in the underlying world. This will not hold true for either lossy or stochastic observation functions. Nonetheless, similar arguments have been made theoretically and empirically

[4]In 1688, William Molyneux posed the question: could someone born blind, upon being given sight, be able to distinguish shapes by vision alone? (Locke 1690) Our arguments suggest an answer: not immediately, but after a bit of visual experience (to form a visual representation) it should be easy (by mapping to prior touch-based representations). Empirical data shows that indeed congentially blind children given sight can quickly learn such abilities (Held, Ostrovsky, Gelder, et al. 2011).

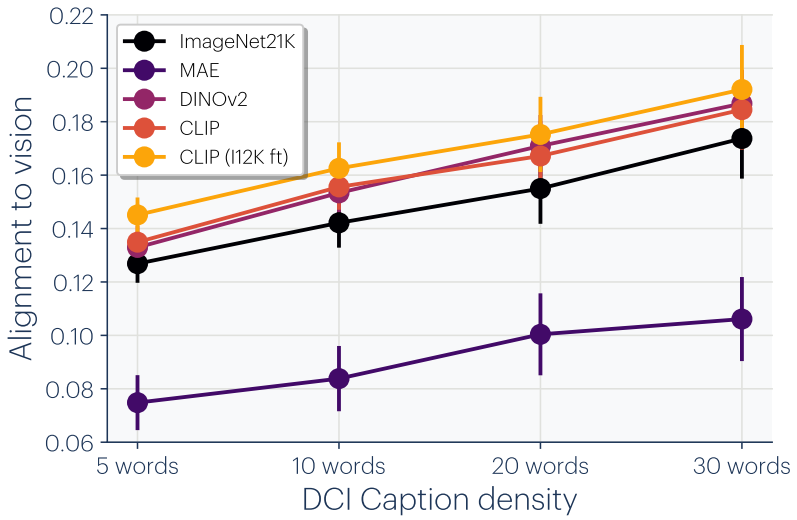


Figure 3.9: **Increasing caption density improves alignment:** We vary caption length using the Densely-Captioned-Images (DCI) dataset (Urbanek, Bordes, Astolfi, et al. 2023). Starting from a dense caption, we used LLaMA3-8B-Instruct (Meta 2024) to summarize and generate coarse-grained captions. We compute the average alignment score across all vision and language models with standard deviation measured over the language models we evaluated. With denser captions, the mapping may become more bijective, leading to improved language-vision alignment scores.

that cooccurrence relations are learned by practical contrastive (Wang and Isola 2020; Zimmermann, Sharma, Schneider, et al. 2021) and predictive learners (Papyan, Han, and Donoho 2020; Roeder, Metz, and Kingma 2021). Lu, Grover, Abbeel, et al. (2021) and Mirchandani, Xia, Florence, et al. (2023) also showed that models trained to autoregressively generate text also capture statistical relations in many other modalities, including symbolic reasoning, vision, protein folding, and robotics.

A more nuanced version of our hypothesis will need to be developed to handle the case of non-bijective observations and abstract concepts. A starting point could be: different models will converge to the same representation *when the input signals are sufficiently high information and the models are sufficiently high capacity*; when they are not, the lower-information representation will only align with the higher-information one up to a level capped by the mutual information between the input signals and by the capacity of each model. This cap might or might not be practically important. Popular representations like CLIP are explicitly optimized to only capture the shared information between vision and language, yet are highly successful on many pure vision tasks.

We perform a preliminary test of the effect of information level in Figure Figure 3.9 (detailed in Appendix Section 6.5), and find that the more descriptive (higher information) a caption is, the better its LLM representation aligns with the visual representation of the corresponding image.

Not all representations are presently converging Our argument has mainly focused on two modalities: vision and language. While we do expect other modalities will follow

similar trends, we have yet to see the same level of convergence across all domains. For example, in robotics there is not yet a standardized approach to representing world states in the same way as there is for representing images and text. One limitation lies in the hardware used in robotics, which is often expensive and slow. This creates a bottleneck in the quantity and diversity of training data.

Sociological bias in producing AI models Researcher bias and collective preferences within the AI community have shaped the trajectory of model development. There is often an explicit or implicit goal of designing AI systems that mimic human reasoning and performance, and this could lead to convergence toward human-like representations even if other kinds of intelligence are in fact possible. Additionally, the "hardware lottery" (Hooker 2021) suggests that the success of AI models can also depend on the compatibility of their design with available computational architectures, further contributing to convergent trends.

Special-purpose intelligences might not converge Different intelligent systems can be designed to accomplish different tasks. For instance: A bioinformatics systems might predict protein structure; an autonomous vehicle might follow lanes on highways. It's possible that not much is shared between these two narrow tasks.

Our argument only holds for intelligences that are optimized to perform well on *many* tasks. We have argued that a representation of *reality* is a structure that is useful across many tasks, but for any special purpose there may be shortcuts, or even effective representations detached from reality. Such shortcuts may be more efficient and necessary for continued improvements in specific domains. This will become more relevant if continued scaling comes up against boundary conditions around resources like energy and compute.

How do we measure alignment? We focused on one particular alignment measure, mutual nearest-neighbor, in our experiments, and cited experiments using several others. However, there is active debate on the merits and deficiencies of all these ways of measuring alignment (Bansal, Nakkiran, and Barak 2021; Sucholutsky, Muttenthaler, Weller, et al. 2023). We discuss our choice and show results for other alignment metrics in Section 6.1.

Lots left to explain We have shown results where different models arrive at *similar* but not the *same* representations. For example, in Figure 3.3, alignment clearly increases but only reaches a score of 0.16, according to our mutual nearest-neighbor metric. The maximum theoretical value for this metric is 1. Is a score of 0.16 indicative of strong alignment with the remaining gap being "noise" or does it signify poor alignment with major differences left to explain? We leave this as an open question.

Chapter 4

An ecosystem for collective intelligence

The preceding chapters have outlined why scaling AI systems appears to be an inevitable path toward the so-called "Platonic representation." The remaining question is: *How do we get there?* Regardless of future hardware advancements, naively scaling up models in a single data center does not seem to be an efficient route to this endpoint. The escalating complexity of state-of-the-art deep learning models presents significant challenges in terms of computational demand, memory requirements, and communication bandwidth. Even as we discover more efficient algorithms, these challenges are likely to persist. Therefore, it is crucial to explore more efficient alternatives.

In light of this, I have advocated for the idea of scaling through collective intelligence. This approach includes:

1. Training a giant monolithic model through the integration of smaller specialized models.
2. Scaling up problem-solving via the coordination of these smaller models.

Both approaches resemble the intelligence seen in human society, where individuals specialize and contribute to a larger ecosystem (*e.g.*, the internet, companies, communities), where the advantages lie in the parallelization of data collection and information processing.

If such a distributed approach proves more cost-effective, it will likely become the preferred path toward general intelligence. However, training or coordinating smaller, specialized models requires new learning paradigms, which the field currently lacks. This chapter proposes one such ecosystem that uses parallel low-rank adapters to train large monolithic models.

4.1 Parallel low-rank distributed training

As computational demands exceed the capacity of modern-day GPUs, training larger models requires innovative solutions. A prominent example of finetuning models is low-rank adaptation (Hu, shen, Wallis, et al. 2022) that uses a low-rank parameterization of a deep neural network to reduce memory requirements for storing optimizer-state and gradient communication during training. The memory requirement was further reduced by quantizing

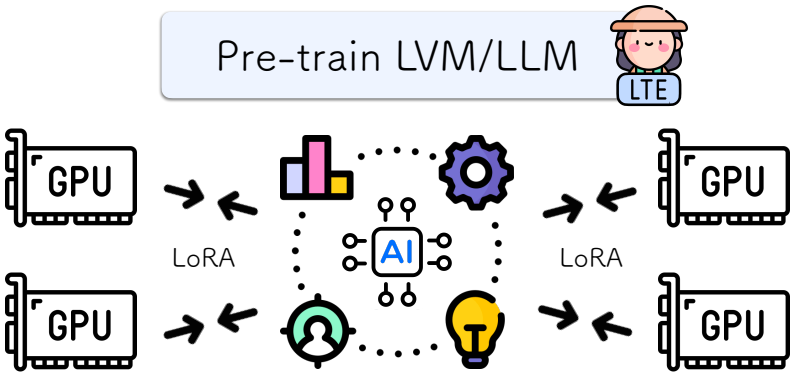


Figure 4.1: **Lora-The-Explorer:** We propose LoRA-the-explorer, an optimization algorithm that can match the performance of standard training from scratch. Our method optimizes unique LoRA parameters in parallel and merges them back to the main weights. Our algorithm can leverage lower-memory devices and only depends on communicating the LoRA parameters for training, making it an ideal candidate in a bandlimited or memory-constraint training framework.

model parameters (Dettmers, Pagnoni, Holtzman, et al. 2023). Such innovations have enabled the finetuning of large models, even on a single consumer-grade GPU. The goal of this chapter is to extend adaptation methods to model pre-training. Specifically, we ask the question: *Can neural networks be trained from scratch using low-rank adapters?*

Successfully addressing this question carries substantial implications, especially considering that common distributed training frameworks are bottlenecked by slow communication speed and bandwidth.

4.1.1 Related works

Training with adapters The use of LoRA has received considerable attention in recent literature (Chavan, Liu, Gupta, et al. 2023; Zhang, Chen, Bukharin, et al. 2023): from reducing computational requirements (Zhang, Zhang, Shi, et al. 2023) to offsetting part of the pre-training computation (Lialin, Shivagunde, Muckatira, et al. 2023).

Of course, LoRA is just one type of adapter, and various forms of adapters have been previously explored. Side-tuning (Zhang, Fang, Zhang, et al. 2021) augment existing models with additional parameters. Adapters in the form of residual connections (Cai, Gan, Zhu, et al. 2020) or affine parameters in normalization layers (Bettelli, Carrier, Gao, et al. 2006; Mudrakarta, Sandler, Zhmoginov, et al. 2018) have also been a popular choice for fine-tuning.

Adapters have been applied to numerous tasks: natural language processing (Houlsby, Giurgiu, Jastrzebski, et al. 2019; Stickland and Murray 2019), video (Yang, Du, Dai, et al. 2024; Xing, Dai, Hu, et al. 2023), computer vision (Sax, Zhang, Zamir, et al. 2020; Zhang and Agrawala 2023; Chen, Duan, Wang, et al. 2023), incremental learning (Rosenfeld and Tsotsos 2018), domain adaptation (Rebuffi, Bilen, and Vedaldi 2018), and vision-language tasks (Gao, Geng, Zhang, et al. 2023; Radford, Kim, Hallacy, et al. 2021; Sung, Cho, and Bansal 2022), text-to-vision generative models (Mou, Wang, Xie, et al. 2023), and even

perceptual learning (Fu, Tamir, Sundaram, et al. 2023).

Distributed training and federated learning Our work shares relevance to both distributed and federated learning paradigms, where each LoRA head can be conceptualized as a distinct computational device. Our work shares many motivations for federated learning, including low-compute devices, high-latency training, privacy, and cross and in-silo learning. See (McMahan, Moore, Ramage, et al. 2017; Wang, Charles, Xu, et al. 2021) for a comprehensive discussion.

Communication efficiency serves as a cornerstone in both distributed and federated learning. Techniques such as *local steps* have been employed to mitigate communication load (McMahan, Moore, Ramage, et al. 2017; Lin, Stich, Patel, et al. 2020; Povey, Zhang, and Khudanpur 2014; Smith, Forte, Chenxin, et al. 2018; Su and Chen 2015; Zhang, De Sa, Mitliagkas, et al. 2016). These methods defer the averaging of weights to specific optimization steps, thus reducing the communication cost per iteration. The effectiveness of decentralized training has been studied in (Lian, Zhang, Zhang, et al. 2017; Koloskova, Stich, and Jaggi 2019; Koloskova, Loizou, Boreiri, et al. 2020; Coquelin, Debus, Götz, et al. 2022).

Traditionally, activation computations have dominated the computational load. However, the advent of gradient checkpointing (Chen, Xu, Zhang, et al. 2016) and reversible gradient computation (Gomez, Ren, Urtasun, et al. 2017; Mangalam, Fan, Li, et al. 2022) has shifted the training process toward being parameter-bound. Techniques such as gradient or weight compression also seek to reduce the communication burden (Lin, Han, Mao, et al. 2018; Aji and Heafield 2017; Wen, Xu, Yan, et al. 2017).

Combining models in federated learning is often credited to FedAvg (McMahan, Moore, Ramage, et al. 2017). Numerous studies explore the use of weighted averaging to improve convergence speed (Li, Huang, Yang, et al. 2020). Since then, many works have tried to use probabilistic frameworks to understand and improve merging (Hsu, Qi, and Brown 2019; Wang, Tantia, Ballas, et al. 2020; Reddi, Charles, Zaheer, et al. 2021). The conditions for optimal merging is still an open question, with recent efforts to improve updating with stale parameters (Chen, Xie, Ma, et al. 2022).

Server momentum and adaptive methods constitute another active area of research where macro synchronization steps are interpreted as "gradients", allowing one to prescribe a bi-level optimization scheme (Hsu, Qi, and Brown 2019; Wang, Tantia, Ballas, et al. 2020; Reddi, Charles, Zaheer, et al. 2021).

Initial efforts have been made to use federated learning with large models. (Yuan, He, Davis, et al. 2022) examined the cost models for pre-training Language Learning Models (LLMs) in a decentralized configuration. (Wang, Lu, Yuan, et al. 2023) suggested the utilization of compressed sparse optimization methods for efficient communication. Our work is also closely related to (Douillard, Feng, Rusu, et al. 2023), a concurrent work that explores the idea of distributing computing across smaller server farms while maintaining low communication costs.

Linear mode connectivity and model averaging Linear mode connectivity (Garipov, Izmailov, Podoprikhin, et al. 2018) studies the phenomena of why and when models are

smoothly connected (Freeman and Bruna 2016; Draxler, Veschgini, Salmhofer, et al. 2018; Fort and Jastrzebski 2019). Under the same initialization, models have been shown to have a linear path with constant energy (Nagarajan and Kolter 2019; Frankle, Dziugaite, Roy, et al. 2020; Wortsman, Ilharco, Gadre, et al. 2022). For models with different initializations, parameter permutations can be solved to align them linearly (Brea, Simsek, Illing, et al. 2019; Tatro, Chen, Das, et al. 2020; Entezari, Sedghi, Saukh, et al. 2021; Simsek, Ged, Jacot, et al. 2021).

Following this line of research, numerous works have also explored model averaging and stitching. Where averaging of large models has shown to improve performance (Wortsman, Ilharco, Gadre, et al. 2022; S. Ainsworth, Hayase, and Srinivasa 2023; Stoica, Bolya, Bjorner, et al. 2023; Jordan, Sedghi, Saukh, et al. 2022; Wortsman, Gururangan, Li, et al. 2022). Model stitching (Lenc and Vedaldi 2015) has also shown to yield surprising transfer capabilities (Moschella, Maiorca, Fumero, et al. 2022). This idea is conceptually related to optimal averaging in convex problems (Scaman, Bach, Bubeck, et al. 2019) and the "Anna Karenina" principle where successful models converge to similar solutions (Bansal, Nakkiran, and Barak 2021). This phenomenon could help explain the success of averaging multiple LoRAs into a single global LoRA (Yi, Yu, Wang, et al. 2023) and averaging MoE MLP layers into a single weight at inference (Wang, Mukherjee, Liu, et al. 2022).

4.1.2 Preliminaries

Unless stated otherwise, we denote x as a scalar, $\mathbf{x}$ a vector, $\mathbf{X}$ a matrix, $\mathcal{X}$ a distribution or a set, $f(\cdot)$ a function and $F(\cdot)$ a composition of functions, and $\mathcal{L}(\cdot, \cdot)$ a loss-function.

Parameter efficient adapters

Adapters serve as trainable functions that modify existing layers in a neural network. They facilitate parameter-efficient finetuning of large-scale models by minimizing the memory requirements for optimization (see Section 4.1.1 for various types of adapters used in prior works).

The focus of this work is on the *low-rank adapter* (Hu, shen, Wallis, et al. 2022, LoRA), a subclass of linear adapters. The linearity of LoRA allows for the trained parameters to be integrated back into the existing weights post-training without further tuning or approximation. Hence, the linearity allows models to maintain the original inference cost. LoRA is frequently used for finetuning transformers, often resulting in less than 10% of the total trainable parameters (even as low as 0.5%).

Low-rank adapter (LoRA) Given input $\mathbf{x} \in \mathbb{R}^n$, and a linear layer $f(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ parameterized by the weight $\mathbf{W} \in \mathbb{R}^{m \times n}$, LoRA re-parameterizes the function as:

$$f_{\mathsf{lora}}(\mathbf{x}) = \mathbf{W}\mathbf{x} + s\mathbf{B}\mathbf{A}\mathbf{x} \tag{4.1}$$

For some low-rank matrices $\mathbf{B} \in \mathbb{R}^{m \times r}$, $\mathbf{A} \in \mathbb{R}^{r \times n}$ and a fixed scalar $s \in \mathbb{R}$, where the rank r is often chosen such that $r \ll \min(m, n)$.

Although the forward pass incurs an extra computational overhead, the significance of LoRA parameterization pertains to the optimizer memory footprint. Optimizers such as AdamW (Kingma and Ba 2015b; Loshchilov and Hutter 2019) typically maintain two states for each parameter, resulting in memory consumption that is twice the size of the trainable parameters. In the case of LoRA parameterization, the optimizer memory scales with the combined sizes of $\mathbf{A}$ and $\mathbf{B}$. This results in significant memory savings when the memory cost of LoRA $\mathcal{O}(r(m+n))$ is less than the memory cost of the model $\mathcal{O}(mn)$. Moreover, QLoRA (Dettmers, Pagnoni, Holtzman, et al. 2023) achieves further memory savings by storing $\mathbf{W}$ in low-precision (4bit) while keeping the trainable parameters $\mathbf{A}$ and $\mathbf{B}$ in higher-precision (16bit). These works have catalyzed the development of several repositories (Wang 2023; Dettmers, Pagnoni, Holtzman, et al. 2023; Dettmers 2023; huggingface 2023), enabling finetuning of models with billions of parameters on low-memory devices.

4.2 Method

To understand the conditions required to pre-train a model with LoRA, we first identify a specific scenario where standard training performance can be recovered using LoRA. This serves as a guide for developing our algorithm that retains the memory efficiency of LoRA.

Although low-rank adapters (LoRAs) have proven to be an effective finetuning method, they have apparent limitations when pre-training. As evidenced in Figure 4.2, models parameterized with LoRA demonstrate inferior performance compared to models trained using standard optimization. This performance gap isn't surprising as it can be attributed to the inherent rank constraint in LoRA. Specifically, for parameter $\mathbf{W} \in \mathbb{R}^{m\times n}$, LoRA is fundamentally incapable of recovering weights that exceed the rank $r < \min(m, n)$. Of course, there are exceptions in which, by happenstance, a solution exists within a low-rank proximity of the initialization. However, in Appendix 8.8, we observed the rank of the gradient tends to increase throughout training, hinting at the necessity for high-rank updates.

4.2.1 Motivation: multi-head merging perspective

This section provides intuition on why LoRA heads in parallel can achieve the performance of standard pre-training.

As demonstrated in Figure 4.2, elevating the rank r of the LoRA to be the same as the rank $\min(m, n)$ of the weight matrix $\mathbf{W} \in \mathbb{R}^{m\times n}$ is sufficient to replicate standard pre-training performance, albeit with different inherent dynamics as detailed in Appendix 8.10. However, such an approach compromises the memory efficiency of low-rank adapters.

Therefore, we investigate the possibility of arriving at an equivalent performance by leveraging multiple low-rank adapters in *parallel*. Our motivation leverages the trivial idea of linearity of these adapters to induce parallelization.

Given a matrix of the form $\mathbf{BA} \in \mathbb{R}^{d_1\times d_2}$ with $\mathbf{B} \in \mathbb{R}^{d_1\times d}$ and $\mathbf{A} \in \mathbb{R}^{d\times d_2}$, it is possible to represent the product as the sum of two lower-rank matrices: $\mathbf{B}_1\mathbf{A}_1 + \mathbf{B}_2\mathbf{A}_2$. To demonstrate this, let $\mathbf{b}_i$ and $\mathbf{a}_i$ be the column vectors of $\mathbf{B}$ and $\mathbf{A}$ respectively. One can then construct

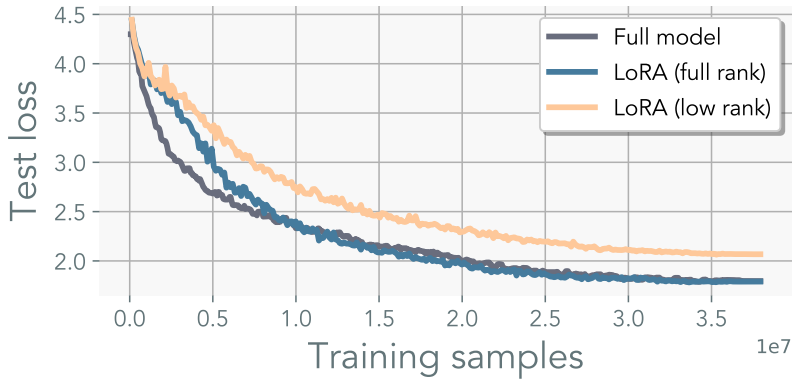


Figure 4.2: **Increasing the rank of LoRA can recover the standard training performance:** ViT-S trained on ImageNet100 using with and without LoRA. Low-rank LoRA uses rank $r = 64$, and full-rank LoRA uses rank $r = \min(m, n)$ set to the dimension of the original weight $\mathbf{W} \in \mathbb{R}^{m \times n}$. Increasing r suffices to match standard training performance.

$\mathbf{B}_1 = [\mathbf{b}_1, \ldots, \mathbf{b}_{[d/2]}]$, $\mathbf{B}_2 = [\mathbf{b}_{[d/2]}, \ldots, \mathbf{b}_d]$, and $\mathbf{A}_1 = [\mathbf{a}_1^T, \ldots, \mathbf{a}_{[d/2]}^T]$, $\mathbf{A}_2 = [\mathbf{a}_{[d/2]}^T, \ldots, \mathbf{a}_d^T]$. This decomposition allows for the approximation of high-rank matrices through a linear combination of lower-rank matrices. The same conclusion can be reached by beginning with a linear combination of rank-1 matrices. This forms the basis for a novel multi-head LoRA parameterization, which we will use as one of the baselines to compare with our final method.

Multi-head LoRA (MHLoRA) Given a matrix $\mathbf{W} \in \mathbb{R}^{m \times n}$, and constant N, multi-head LoRA parameterizes the weights as a linear combination of N low-rank matrices $\mathbf{B}_n$ and $\mathbf{A}_n$:

$$f_{\text{mhlora}}(\mathbf{x}) = \mathbf{W}\mathbf{x} + \frac{s}{N} \sum_{n=1}^{N} \mathbf{B}_n \mathbf{A}_n \mathbf{x} \tag{4.2}$$

Multi-head LoRA reparameterizes the full-rank weights into a linear combination of low-rank weights.

Now, we will point out a trivial observation that a single parallel LoRA head can approximate the trajectory of a single step of the multi-head LoRA, provided that the parallel LoRA heads are periodically merged into the full weights.

Using the same rank r for all the LoRA parameters, the dynamics of a single parallel LoRA head (denoted with $\hat{\cdot}$) is equivalent to multi-head LoRA:

$$\arg\min_{\mathbf{B}_n \mathbf{A}_n} \mathcal{L}\left(\mathbf{W} + \frac{s}{N} \sum_{n=1}^{N} \mathbf{B}_n \mathbf{A}_n\right) = \arg\min_{\hat{\mathbf{B}}_n, \hat{\mathbf{A}}_n} \mathcal{L}\left(\hat{\mathbf{W}} + \frac{s}{N} \hat{\mathbf{B}}_n \hat{\mathbf{A}}_n\right)$$

When either $\sum_{n=1}^{N} \mathbf{B}_n \mathbf{A}_n$ is equal to $\hat{\mathbf{B}}_n \hat{\mathbf{A}}_n$, or when $\hat{\mathbf{W}} = \mathbf{W} + \frac{s}{N} \sum_{j \neq n}^{N} \mathbf{B}_j \mathbf{A}_j$; here we used a shorthand notation to indicate that sum is over all the LoRA parameters except for index n. We assume the parameters on both sides of the equation are initialized to be the same:

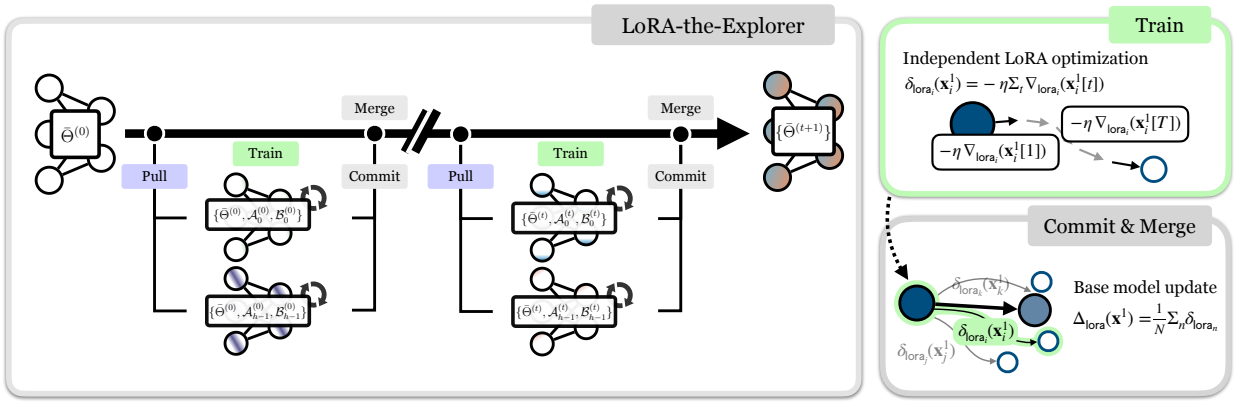


Figure 4.3: **LTE diagram**: Our method is decomposed into 3 steps. (**1**) We parameterize the model with multiple LoRA heads and train them independently for T iterations using different mini-batches sampled from the same (homogeneous) distribution. This results in overall update of $\delta_{\mathsf{lora}_n}(\mathbf{x}) = -\eta \sum_t \nabla_{\mathsf{lora}_n}(\mathbf{x}[t])$ (**2**). Next, we accumulate the individual LoRA updates by averaging the heads $\Delta_{\mathsf{lora}}(\mathbf{x}) = \frac{1}{N}\sum_n \delta_{\mathsf{lora}_n}(\mathbf{x})$. (**3**) The update is applied to the main weights, and the LoRA parameter $\mathbf{B}$ is reset. The optimization repeats with the new LoRA parameters.

$\mathbf{A}_n = \hat{\mathbf{A}}_n$ and $\mathbf{B}_n = \hat{\mathbf{B}}_n \forall n$. The first scenario is rank deficient, which we know is unable to recover the original model performance. The latter case necessitates that $\hat{\mathbf{W}}$ accumulates all the information of the LoRA parameters at every iteration. Hence, if we can apply a merge operator at every iteration, we can recover the exact update.

This rather simple observation implies that one can recover the exact gradient updates of the multi-head LoRA parameterized model, which we observed to match pre-training performance across a wide range of tasks (see Appendix 8.13). Moreover, in a distributed setting, only the LoRA parameters/gradients have to be communicated across devices, which is often a fraction of the original model size, making it a good candidate where interconnect speed between computing nodes is limited.

4.2.2 LoRA soup: delayed LoRA merging

To further reduce the communication cost of LTE, we extend and combine the ideas of local updates (McMahan, Moore, Ramage, et al. 2017) and model-averaging (Wortsman, Ilharco, Gadre, et al. 2022; Yadav, Tam, Choshen, et al. 2023; Ilharco, Ribeiro, Wortsman, et al. 2023). Instead of merging every iteration, we allow the LoRA parameters to train independently of each other for a longer period before the merge operator. This is equivalent to using stale estimates of the LoRA parameters $\hat{\mathbf{W}} = \mathbf{W} + \frac{s}{N}\sum_{j\neq n}^{N} \mathbf{B}'_j\mathbf{A}'_j$ with $'$ indicating a stale estimate of the parameters.

Merging every iteration ensures that the representation will not diverge from the intended update. While using stale estimates relaxes this equivalence, we observe that it can still match the standard training performance as shown in Table 4.9. Nevertheless, as the estimate becomes inaccurate, the optimization trajectory does indeed diverge from the optimization path of multi-head LoRA. We quantify this divergence in Figure 4.4. The divergence does

not imply that the model won't optimize; rather, it suggests that the optimization trajectory will deviate from that of the multi-head LoRA. In this work, we opt for simple averaging and leave more sophisticated merging such as those used in (Karimireddy, Kale, Mohri, et al. 2020; Matena and Raffel 2022; Yadav, Tam, Choshen, et al. 2023) for future works.

4.2.3 LoRA-the-Explorer: parallel low-rank updates

Our algorithm is designed with two primary considerations: (1) achieving an informative update $\Delta\mathbf{W}$ that does not require materialization of the full parameter size during training, and (2) parameterizing $\mathbf{W}$ such that it can be stored in low-precision and communicated efficiently. The latter can be achieved by using quantized weights and keeping a high-precision copy of $\mathbf{W}$.

We propose LoRA-the-Explorer (LTE), an optimization algorithm that approximates full-rank updates with parallel low-rank updates. The algorithm creates N-different LoRA parameters for each linear layer at initialization. Each worker is assigned the LoRA parameter and creates a local optimizer. Next, the data is independently sampled from the same distribution $\mathbf{x} = \{\mathbf{x}_1, \dots \mathbf{x}_N\}$. For each LoRA head n, the parameters are optimized with respect to its own data partition for T iterations resulting in an update $\delta_{\mathsf{lora_n}} = -\eta \sum_{t=1}^{T} \nabla_{\mathsf{lora}_n} \mathbf{x}_i[t]$. We do not synchronize the optimizer state across workers. After the optimization, the resulting LoRA parameters are synchronized to compute the final update for the main weight $\Delta_{\mathsf{lora}}(\mathbf{x}) = \frac{1}{N} \sum_{n=1}^{N} \delta_n$. In the next training cycle, the LoRA parameters are trained with the updated weights $\mathbf{W}$. Here, the LoRA parameters can be either be re-initialized or the same parameters can be used with the correction term (see Appendix 8.2). Since we do not train directly on the main parameter $\mathbf{W}$, we can use the quantized parameter $q(\mathbf{W})$ instead. Where one can either keep the high-precision weight only in the master node or offload it from the device during training. This reduces not only the memory footprint of each worker but also the transmission overhead. An illustration in Figure 4.3.

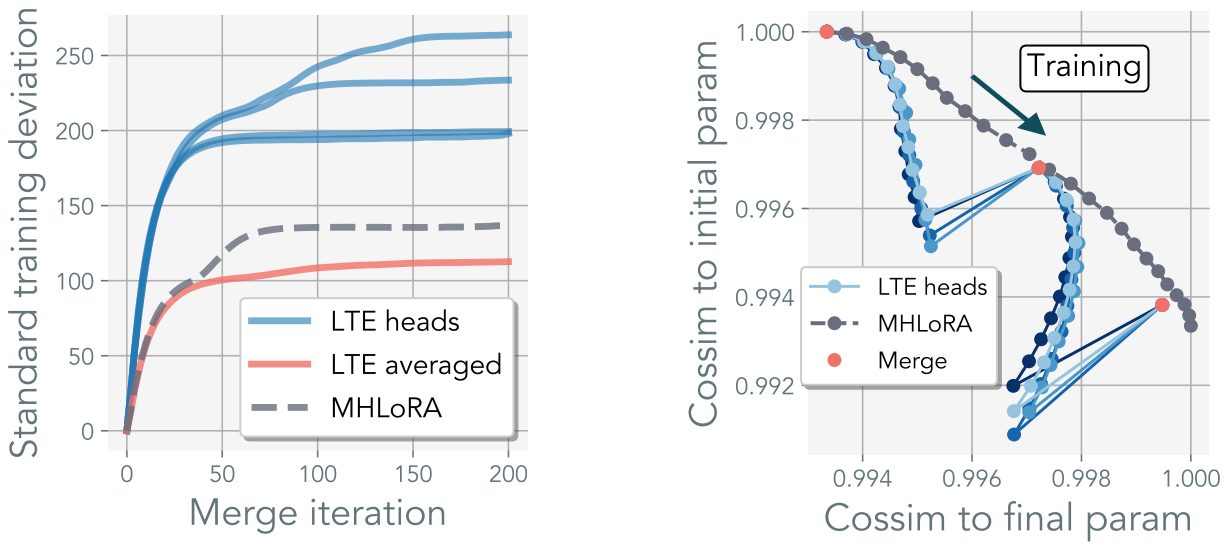
4.3 Experiments

We follow standard training protocols, and all implementation details and training hyper-parameters can be found in Appendix 8.1.

Disclaimer: During the preparation of the code release, we found that the transformer experiments used a scaling factor of $1/\sqrt{d_{out}}$ instead of the standard scaling $1/\sqrt{d_{out}/n_{attn}}$ (Vaswani, Shazeer, Parmar, et al. 2017). Hence using LTE with the standard scaling would require a different set of hyper-parameters than the ones reported in Appendix 8.1. We will revise the hyper-parameters in the next revision.

4.3.1 Iterative LoRA Merging

In Section 4.2.1, we motivated that iteratively merging LoRA parameters is a key component in accurately recovering the full-rank representation of the model. As a sanity check, in Appendix 8.3, we assess the effectiveness of merging a single LoRA head in the context



(a) Gradient deviation

(b) Trajectory projection

Figure 4.4: **Effects of merging LoRA heads. Left:** We measure l_2-norm deviation of the effective weights of multi-head LoRA (MHLoRA) and LoRA-the-explorer (LTE, our method) from the weights of standard training using ViT-S. We use 4 heads for both MHLoRA and LTE using the same initialization, and we measure the norm of `encoder-layer-3`. We also plot the individual LoRA heads of LTE. These heads deviate more from standard training, but their average closely follows that of MHLoRA. Depending on the merge iteration (x-axis), the estimation gap of using stale estimates is roughly the difference between the MHLoRA and LTE averaged. The later the merge happens, the more LTE deviates from MHLoRA. **Right:** We project the dynamics of MHLoRA and LTE onto the parameters of MHLoRA. The y-axis is the initial parameters, and the x-axis is after training for 25 iterations. The projection is computed by computing the cosine similarity on the vectorized weights and creating an arc from (0, 1) to (1, 0). We set merge iteration to 12 and visualize how the LTE trajectory follows the arc of MHLoRA.

of linear networks trained on synthetic least-squares regression datasets. The underlying rank of the optimal solution, $\mathbf{W}^*$, is controlled, and datasets are generated as $\mathbf{Y} = \mathbf{X}(\mathbf{W}^*)^\mathsf{T}$. Each $\mathbf{x} \in \mathbf{X}$ follows a normal distribution, $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Figure 8.1 evaluates the model's rank recovery across varying merge iteration T. Dimension of the weights $\mathbf{W}$ are set to $m = n = 32$. Without merging, the model performance plateaus rapidly on full-rank $\mathbf{W}^*$. In contrast, iterative merging recovers the ground truth solution with the rate increasing with higher merge frequency.

Further tests in Figure 4.5 using ViT-S (Dosovitskiy, Beyer, Kolesnikov, et al. 2020) with a patch-size of 32 on the ImageNet100 dataset (Tian, Krishnan, and Isola 2020) (a subset of ImageNet (Russakovsky, Deng, Su, et al. 2015)) confirm that merging of a single LoRA head outperforms standalone LoRA parameter training. However, frequent merging delays convergence, likely due to LoRA parameter re-initialization and momentum state inconsistencies. Additionally, the performance does not match that of fully trained models, indicating potential local minima when training with rank-deficient representations.

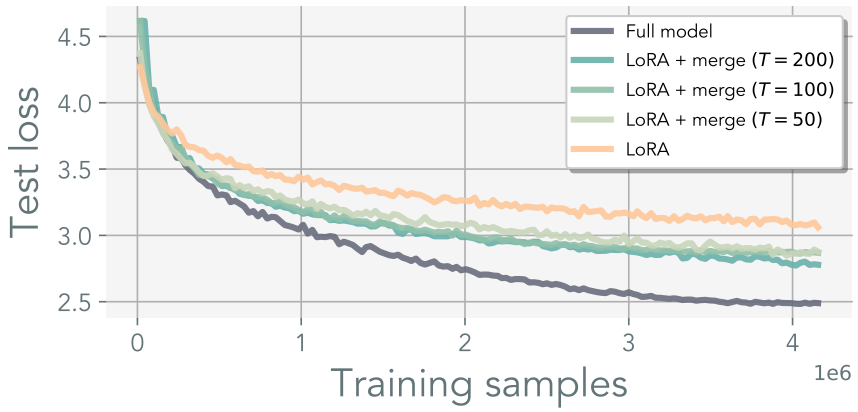


Figure 4.5: **Sequential merging of LoRA cannot recover performance.** ViT-S trained on ImageNet100. Merging and resetting the LoRA parameters achieves better performance than single-head LoRA pre-training but still cannot recover the standard full-model pre-training. Note that LoRA + merge is akin to concurrent work of ReLoRA (Lialin, Shivagunde, Muckatira, et al. 2023).

We find that the merge iteration of $T = 10$ is still stable when using batch size 4096. With higher T values, additional training may be required to achieve comparable performance (Stich 2019; Wang and Joshi 2021; Yu, Yang, and Zhu 2019). Our initial efforts to improve merging using methods such as (Yadav, Tam, Choshen, et al. 2023) did yield better results. Nonetheless, we believe with increased merge iteration, smarter merging techniques may be necessary. Existing literature in federated learning and linear-mode connectivity/model-averaging may provide insights in designing better merging criteria.

To further test the generalizability of our method, we conducted a suite of experiments on various vision tasks in Figure 4.6. Moreover, we test our method on MLP-Mixer (Tolstikhin, Houlsby, Kolesnikov, et al. 2021) to demonstrate its use outside of transformer architecture. We provide additional experiments when using $T = 1$ and initial language-modeling results in Appendix 8.13.

4.3.2 LoRA parameter alignment

The efficacy of our optimization algorithm hinges on the ability of individual heads to explore distinct subspaces within the parameter space. We examine the extent to which data and initial parameters influence intra-head similarity throughout training. In Figure 4.7, we compute the average cosine similarity and Grassman distance (see Appendix 8.5) between the heads $\mathbf{B}_n\mathbf{A}_n$. These tests were conducted with data samples drawn from the same distribution, and each set of LoRA parameters was exposed to a different set of samples.

Our results confirm that LoRA heads do not converge to the same representation. We find using different initializations across LoRA heads yields the greatest orthogonality. This orthogonality is further increased when different mini-batches are used for each head. Importantly, the degree of alignment among LoRA heads remains stable post-initialization and does not collapse into the same representation. In Figure 4.7, we find that lower similarity

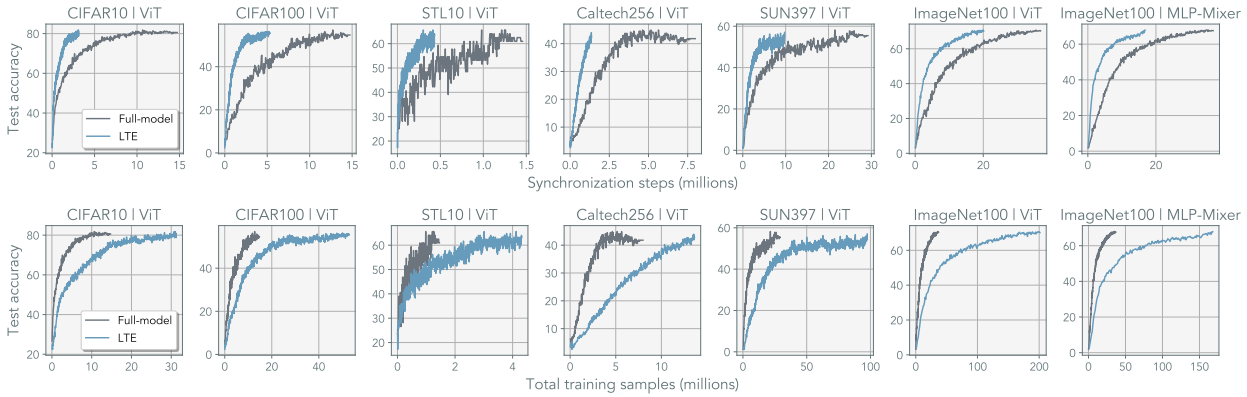


Figure 4.6: **Experiments on various vision tasks**: We apply our method to a range of vision tasks, including CIFAR10, CIFAR100 (Krizhevsky, Hinton, et al. 2009), STL10 (Coates, Ng, and Lee 2011), Caltech256 (Griffin, Holub, and Perona 2007), SUN397 (Xiao, Hays, Ehinger, et al. 2010), and ImageNet100 (Tian, Krishnan, and Isola 2020). Details of these datasets are listed in Appendix 8.1. Additionally, we incorporated the MLP-mixer and observed consistent results. The figure is presented with two different x-axes. On the top, we plot the total number of synchronization steps, and on the bottom, we measure the total number of training samples observed across all devices. All LTE experiments are conducted on ViT-S with a merge iteration of $T = 10$ and 32 LoRA heads with a rank of $r = 64$. In Appendix 8.13, we provide additional results for $T = 1$, LLM experiments, and LTE trained on ViT and MLP-mixer at various scales.

corresponds well with model performance, where using different parameters and mini-batches significantly outperforms other configurations.

4.3.3 The effect of LoRA heads, rank, and merge iteration

We systematically evaluate the effects of varying the number of LoRA heads, rank, and merge iteration on model performance for ImageNet100 in Table 4.9. Our findings indicate a monotonic improvement in performance with an increased number of heads and ranks. Conversely, extending the merge iteration negatively impacts performance. As in the case of least-squares regression, we found excessive merging to hurt model accuracy. With a large enough rank and head, we found the model to converge to better test accuracy, even if the test loss was similar. We hypothesize the averaging of the LoRA heads has a regularization effect similar to that model ensembling.

We use ViT-S as the primary architecture for analysis, which has a hidden dimension of 384 and an MLP dimension of 1536. We find that setting the product of the number of heads and the rank of the LoRA larger than the largest dimension of the model serves as a good proxy for configuring LTE. For example, using 32 heads with $r = 64$ results in $2048 > 1536$. However, when it comes to increasing the number of heads rather than rank, we noticed longer training iterations were required to achieve comparable performance. We discuss a potential cause of the slowdown in convergence in the preceding section.

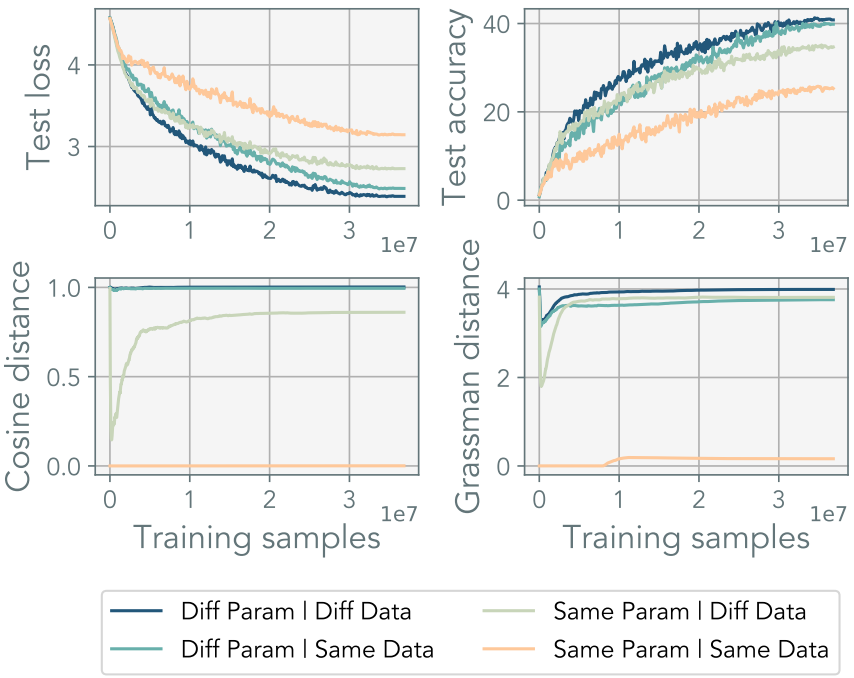


Figure 4.7: **LoRA alignment**: Alignment of LTE heads when varying parameters and data. "Diff Param" uses random initialization across each head, and "Diff Data" uses different mini-batches. The similarity is computed on the first epoch of ImageNet100 on ViT-S. We use LTE of r=8 with 4 LoRA heads. Pair-wise similarity is averaged across all linear layers. The performance of the model correlates with orthogonality between LoRA heads.

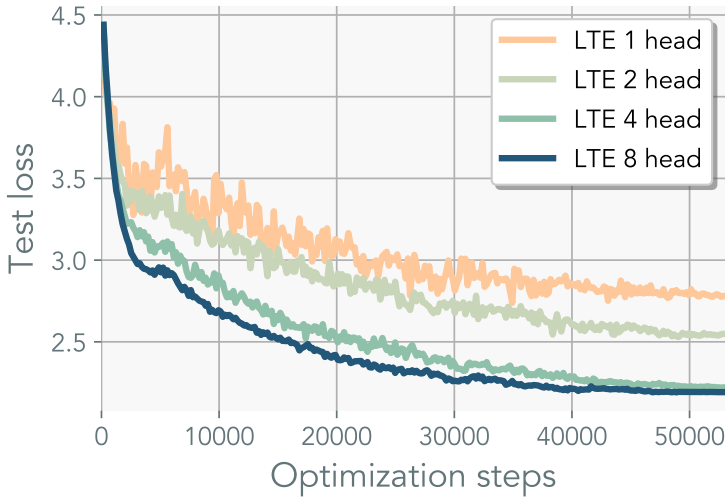


Figure 4.8: **LTE with same batch-size per head:** ViT-S trained on ImageNet100. We use the same batch-size for each LoRA head with rank $r = 8$. In contrast to other figures, we plot the loss in optimization steps. LTE with more heads converge to a better solution but require longer training samples to converge.

	LTE	Heads	Rank	Merge	Test loss ↓	Test acc ↑
	-	-	-	-	1.78	71.97
head	✓	2	8	10	2.31	54.68
	✓	8	8	10	2.35	54.88
	✓	32	8	10	2.26	58.21
	✓	2	64	10	1.86	69.82
	✓	8	64	10	1.84	71.97
	✓	32	64	10	1.79	73.73
rank	✓	32	8	10	2.66	44.00
	✓	32	16	10	2.12	60.35
	✓	32	32	10	1.81	71.20
	✓	32	64	10	1.79	73.73
	✓	32	128	10	1.78	73.54
merge	✓	32	64	5	1.87	70.67
	✓	32	64	10	1.79	73.73
	✓	32	64	20	1.97	66.80
	✓	32	64	50	1.99	65.43
	✓	32	64	100	2.10	61.04

Figure 4.9: **LTE ablation for ViT-S trained on ImageNet100:** For *fixed cumulative training epoch* of 1200, we vary the number of heads, rank, and merge iteration of our method. More heads require longer cumulative training samples to converge; see Figure 4.8.

4.3.4 Gradient noise with parallel updates

In our ablation study, we utilized a fixed cumulative batch size of 4096 and a training epoch of 1200. Each LoRA head received a reduced batch size of $\frac{4096}{\text{heads}}$. Our findings indicate that scaling the rank exerts a greater impact than increasing the number of heads. Due to the proportional scaling of gradient noise with smaller mini-batches (McCandlish, Kaplan, Amodei, et al. 2018; Shallue, Lee, Antognini, et al. 2019; Smith and Le 2018), we hypothesize that gradient noise is the primary factor contributing to slower convergence, in addition to the use of stale parameter estimates. To validate this hypothesis, we employed the same mini-batch size across all heads in Figure 4.8, using a reduced rank of $r = 8$. When we adjusted the batch size in proportion to the number of heads and measured it with respect to the optimization steps, the impact of varying the number of heads became more pronounced. While increasing the number of heads necessitates more sequential FLOPs, it offers efficient parallelization. Furthermore, using a larger batch size for gradient estimation may prove beneficial in distributed training, as it increases the computational workload on local devices. Careful optimization of the effective batch size to maximize the signal-to-noise ratio may be crucial for achieving maximum FLOP efficiency.

4.3.5 Performance Scaling on ImageNet-1K

We scaled up our method to ImageNet-1K. We followed the training protocols detailed in Appendix 8.1. In accordance with our initial hypothesis on gradient noise, we doubled the batch size to 8192 (see Appendix 8.7). Since using different mini-batches was crucial early in training, we did not alter the way mini-batches were sampled. Scheduling the randomness

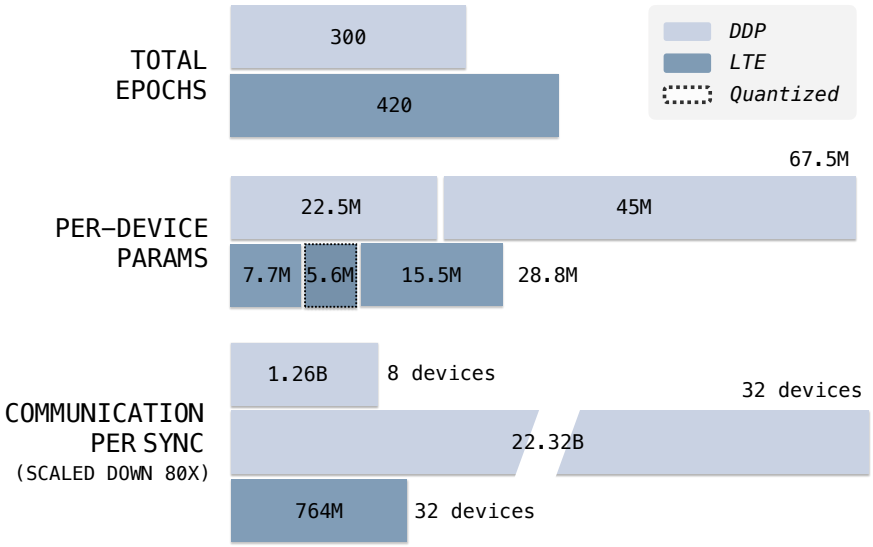


Figure 4.10: **ImageNet compute analysis**: We break down the computational cost for training ViT-S on ImageNet1k. We compare against distributed data-parallel with 8 devices. LTE requires 40% longer to achieve the same performance of 68% top-1 accuracy on 1000-way classification. We used 32 LoRA heads with $r = 64$. Our method requires fewer trainable parameters per device. This, in turn, enables the fitting of larger models in low-memory devices. With a smaller memory footprint and infrequent communication, our method requires lower communication bandwidth. Further discussion is in Section 4.3.5.

for the mini-batches is an option we have not yet explored.

In the initial training phase, we observed that LTE outperformed standard training. However, as training approached completion, standard training overtook LTE, necessitating additional iterations for LTE to achieve comparable performance. Standard training appeared to benefit more from a lower learning rate compared to LTE. For ViT-S, the model took 40% more training samples to converge to the same top-1 accuracy of 68% (see Appendix 8.6).

The primary focus of our work was to investigate whether it is possible to train deep networks with parallel low-rank adapters; hence, we did not aim to maximize efficiency. However, we do provide a hypothetical computation analysis for future scaling efforts. Let the model size be denoted by $M_{\text{ddp}} = M$, and M_{lte} for LTE, and the respective number of devices for each method be denoted with N_{ddp}, and N_{lte}. With quantization, each LTE device would require a memory footprint of $qM + M_{\text{lte}}$. With the base model operating in 16-bit precision, using 4-bit quantization results in $q = 0.25$. With AdamW, DDP necessitates an additional $2M$ parameters, making the total memory footprint $3M$ per device. For LTE, the total memory footprint per device is $qM + 3M_{\text{lte}}$. Assuming the training is parameter-bound by the main weights $r \ll \min(m, n)$, LTE can leverage GPUs that are roughly 1/3 the size required for DDP. It is worth noting that LTE requires 40% more data to train and a slowdown of 20% per iteration when using quantization methods such as QLoRA. If the cost of low-memory devices is lower, these slowdowns may be negligible compared to the speed-up achieved through parallelization. On average, each LTE device observes 1/3 less data than a device in DDP. With improvements in our method and future advances in quantization, we believe this gap

will be reduced. The compute analysis for ViT-S on ImageNet1k is illustrated in Figure 4.10.

Communication also presents bottlenecks when training models across nodes. For a single node, one can interleave the communication of gradients asynchronously with the backward pass (Li, Zhao, Varma, et al. 2020). In multi-node systems, the communication scales with the size of the trained parameters and is bottlenecked at interconnect speed, especially when high-throughput communication hardware, such as InfiniBand, is not utilized. Using standard all-reduce, the gradient is shared between each device for a total communication of $N_{\mathsf{ddp}}(N_{\mathsf{ddp}}-1)M$. For LTE we communicate every T iteration hence we have $\frac{1}{T}N_{\mathsf{lte}}(N_{\mathsf{lte}}-1)M$. To maximize the efficacy of LTE, an alternative approach is to use a parameter server for 1-and-broadcast communication. Here, gradients are sent to the main parameter server and averaged. The accumulated updates are broadcast back to other nodes. DDP with a parameter server would use $2(N_{\mathsf{ddp}}-1)M$ and LTE would use $\frac{1}{T}((N_{\mathsf{lte}}-1)M_{\mathsf{lte}}+(N_{\mathsf{lte}}-1)qM)$. Moreover, LTE can leverage lower-bandwidth communication since the parameters shared between devices are strictly smaller by a factor of $M_{\mathsf{ddp}}/M_{\mathsf{lte}}$.

4.4 Discussion

This work investigated the feasibility of using low-rank adapters for model pre-training. I introduced LTE, a bi-level optimization method that capitalizes on the memory-efficient properties of LoRA. Although we succeeded in matching performance on moderately sized tasks, several questions remain unresolved. These include: how to accelerate convergence during the final 10% of training; how to dynamically determine the number of ranks or heads required; whether heterogeneous parameterization of LoRA is feasible, where each LoRA head employs a variable rank r; and leveraging merging strategies to accompany higher local optimization steps. Our work serves as a proof-of-concept, demonstrating the viability of utilizing low-rank adapters for neural network training from scratch. However, stress tests on larger models are essential for a comprehensive understanding of the method's scalability. Addressing these open questions will be crucial for understanding the limitations of our approach. We anticipate that our work will pave the way for pre-training models in computationally constrained or low-bandwidth environments, where less capable and low-memory devices can collaboratively train a large model, embodying the concept of the "*wisdom of the crowd.*"

Chapter 5

Epilogue

Throughout this thesis, we have explored the phenomenon of convergence in modern AI systems. The key insights uncovered are as follows:

In Chapter 2, we examined the inherent preference of deep neural networks for simple, low-rank solutions. By analyzing the effective rank of the learned kernels, we demonstrated that neural architectures have an inductive bias toward parsimonious explanations of data. This simplicity bias appears to be a fundamental characteristic of these models, regardless of the specific architecture or training regime.

Building on this finding, Chapter 3 presented the Platonic Representation Hypothesis—the idea that as AI systems scale, they will converge toward a common representation that models co-occurrences of events in reality. The empirical evidence of increasing model similarity across domains supports the notion that there may indeed be a Platonic "ideal" way to model the world, which multiple systems independently discover.

Finally, Chapter 4 explored potential pathways to efficiently scale up AI systems and reach this hypothetical representational endpoint. Recognizing the limitations of monolithic scaling, we proposed a distributed approach utilizing parallel low-rank adapters. This ecosystem of specialized models mirrors the structure of human intelligence and could be a more viable route to general AI.

The overarching theme of this work is the idea that the pursuit of simplicity is a fundamental driver of both biological and artificial intelligence. Whether it manifests as the preference for low-rank solutions, the convergence toward common representations, or the organization of intelligence into distributed, collaborative systems, the drive toward parsimony appears to be a universal principle underlying intelligent behavior.

As we continue to push the boundaries of what AI can achieve, it will be crucial to understand how to better characterize the progress of artificial systems. This thesis aimed to provide a step in that direction, but there remains much more to explore in this fascinating frontier of science.

Chapter 6

Chapter 2 Appendix

6.1 Mutual *k*-Nearest Neighbor Alignment Metric

For two models with representations f, g the mutual k-nearest neighbor metric measures the average overlap of their respective nearest neighbor sets. In this section, we refer to this metric as $m_{\texttt{NN}}$, which we will formally define below.

For cross-modal domains, define $(x_i, y_i) \in \mathcal{X}$ as a sample from the data distribution $\mathcal{X}$ (*e.g.* image-caption dataset). For the single domain alignment measurements, the samples are equivalent $x_i = y_i$ (*e.g.*, images for vision, and text for language). Let $\{x_i, y_i\}_{i=1}^{b}$ be the corresponding mini-batch sampled from this data distribution. Then given two model representations f and g the corresponding features are: $\phi_i = f(x_i)$ and $\psi_i = g(y_i)$, where the collection of these features are denoted as $\Phi = \{\phi_1, \ldots, \phi_b\}$ and $\Psi = \{\psi_1, \ldots, \psi_b\}$. Then for each feature pair (ϕ_i, ψ_i), we compute the respective nearest neighbor sets $\mathcal{S}(\phi_i)$ and $\mathcal{S}(\psi_i)$.

$$d_{\mathsf{knn}}(\phi_i, \Phi \setminus \phi_i) = \mathcal{S}(\phi_i) \tag{6.1}$$

$$d_{\mathsf{knn}}(\psi_i, \Psi \setminus \psi_i) = \mathcal{S}(\psi_i) \tag{6.2}$$

where d_{knn} returns the set of indices of its k-nearest neighbors. Then we measure its average intersection via

$$m_{\texttt{NN}}(\phi_i, \psi_i) = \frac{1}{k} |\mathcal{S}(\phi_i) \cap \mathcal{S}(\psi_i)| \tag{6.3}$$

where $|\cdot|$ is the size of the intersection.

The choice to use mutual nearest-neighbors Our initial efforts to measure alignment with CKA revealed a very weak trend of alignment between models, even when comparing models within their own modality. This has also been observed by Bansal, Nakkiran, and Barak 2021, which had relied on alternative metrics such as model-stitching as it "reveals aspects of representations that measures such as centered kernel alignment (CKA) cannot" Bansal, Nakkiran, and Barak 2021.

We chose to use nearest-neighbor as a metric, as methods like CKA has a very strict definition of alignment, which may not fit our current needs. For instance, understanding the precise similarity between unrelated items, such as an orange and Bill Gates, may not be critical.

Relationship between CKA and Mutual Nearest-Neighbors Let $\phi_i \in \mathbb{R}^n$ and $\psi_i \in \mathbb{R}^m$ be vectorized features of two models (*e.g.* language and vision models). Let $\mathbf{K}_{ij} = \kappa(\phi_i, \phi_j)$ and $\mathbf{L}_{ij} = \kappa(\psi_i, \psi_j)$ be the kernel matrices computed from a dataset using some kernel-function κ. Using an inner-product kernel, the ij-th entry of the centered counterpart of these Kernel matrices is:

$$\bar{\mathbf{K}}_{ij} = \langle \phi_i, \phi_j \rangle - \mathbb{E}_l[\langle \phi_i, \phi_l \rangle] \qquad \bar{\mathbf{L}}_{ij} = \langle \psi_i, \psi_j \rangle - \mathbb{E}_l[\langle \psi_i, \psi_l \rangle] \tag{6.4}$$

Then, the cross-covariance of $\mathbf{K}$ and $\mathbf{L}$ is given by:

$$\mathsf{HSIC}(\mathbf{K}, \mathbf{L}) = \frac{1}{(n-1)^2} \mathsf{Trace}(\bar{\mathbf{K}}\bar{\mathbf{L}}) \tag{6.5}$$

which serves as an empirical estimator of the Hilbert-Schmidt Independence Criterion Gretton, Bousquet, Smola, et al. 2005. The Centered Kernel Alignment (CKA) Kornblith, Norouzi, Lee, et al. 2019 is then its normalized counterpart:

$$\mathsf{CKA}(\mathbf{K}, \mathbf{L}) = \frac{\mathsf{HSIC}(\mathbf{K}, \mathbf{L})}{\sqrt{\mathsf{HSIC}(\mathbf{K}, \mathbf{K})\mathsf{HSIC}(\mathbf{L}, \mathbf{L})}} \tag{6.6}$$

CKA measures the congruence between two random variables, with a maximum alignment of 1 and a minimum of 0. It is invariant to isotropic scaling and offers a strict notion of alignment, measuring alignment across all samples. Hence, the CKA score reflects the global similarities of the models. This can be illustrated by expanding the trace term in HSIC:

$$\mathsf{Trace}(\bar{\mathbf{K}}\bar{\mathbf{L}}) = \sum_i \sum_j \left(\langle \phi_i, \phi_j \rangle - \mathbb{E}_l[\langle \phi_i, \phi_l \rangle]\right) \left(\langle \psi_i, \psi_j \rangle - \mathbb{E}_l[\langle \psi_i, \psi_l \rangle]\right) \tag{6.7}$$

One can modify the definition of alignment to restrict the cross-covariance measurement to samples considered to be nearest neighbors of the current sample i. This emphasizes similarity over dissimilarity, biasing the measure toward local alignment:

$$\mathsf{Align}_{\mathsf{knn}}(\mathbf{K}, \mathbf{L}) = \sum_i \sum_j \alpha(i,j) \cdot \left(\langle \phi_i, \phi_j \rangle - \mathbb{E}_l[\langle \phi_i, \phi_l \rangle]\right) \left(\langle \psi_i, \psi_j \rangle - \mathbb{E}_l[\langle \psi_i, \psi_l \rangle]\right) \tag{6.8}$$

$$\text{where} \qquad \alpha(i,j) = \mathbb{1}[\phi_j \in \mathsf{knn}(\phi_i) \wedge \psi_j \in \mathsf{knn}(\psi_i) \wedge i \neq j] \tag{6.9}$$

Where $\alpha(i,j)$ is a scalar weighting that assigns 1 if j is a mutual nearest neighbors to both ϕ_i and ψ_i, and 0 otherwise. We refer to this metric as the Centered Kernel Nearest-Neighbor Alignment (CKNNA) metric. As the number of nearest neighbors $k \to \dim(\mathbf{K})$, we recover the original CKA metric.

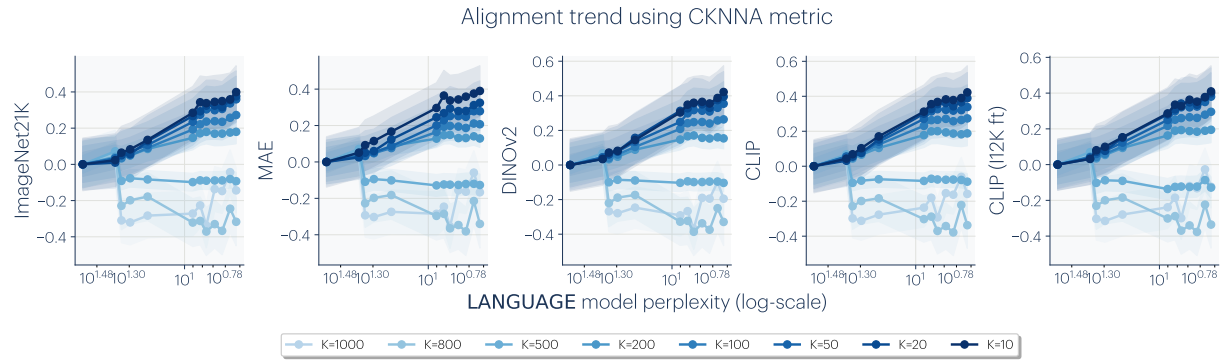


Figure 6.1: **Cross-modal alignment increases locally:** Alignment trend when varying the top-k nearest neighbors in the CKNNA metrics (Eqn. 6.10). We center alignment score to the smallest language model and divide the total trend by the standard deviation. When $k = 1024$, we recover the original CKA metric, and when $k < |\mathcal{X}|$ it closely resembles the mutual nearest-neighbor metric $m_{\texttt{NN}}$. Each line represents the average of all LLM models for a specific k. As we decrease k, the alignment becomes more pronounced.

$$\mathsf{CKNNA}(\mathbf{K}, \mathbf{L}) = \frac{\mathsf{Align}_{\mathsf{knn}}(\mathbf{K}, \mathbf{L})}{\sqrt{\mathsf{Align}_{\mathsf{knn}}(\mathbf{K}, \mathbf{K}), \mathsf{Align}_{\mathsf{knn}}(\mathbf{L}, \mathbf{L})}} \tag{6.10}$$

We can further relax the metric to treat the cross-covariance term identically across all nearest-neighbor samples. This is equivalent to the assumption that all nearby samples have the same distance. This simplification leads us back to the mutual nearest neighbor metric:

$$\sum_i \sum_j \alpha(i, j) \cdot 1 = n \cdot k \cdot m_{\texttt{NN}}(\phi_i, \psi_i) \tag{6.11}$$

By equating these metrics, we analyze the changes in alignment between language and vision models as we vary the number of neighbors k in Eqn. 6.10. In Figure 6.1, we compute the average alignment score across all LLM models. For each k, we center the scores to the smallest vision model and divide by the standard deviation of the scores. We find that high values of k show less conclusive alignment across tasks while decreasing k shows a coherent trend across both models and tasks. We find that certain visual tasks, such as CLIP, exhibit global alignment, whereas methods like ImageNet-21k classification show only local alignment. This observation suggests that cross-modal alignment occurs locally across most common visual tasks, and global alignment may require additional language grounding, as done in the CLIP objective.

6.2 Consistency across various metrics

We describe the metrics in Table 6.2 and their corresponding properties. The *symmetric* property implies that the metric is symmetric with respect to the data points $d(x, y) = d(y, x)$. The *global* property means all samples are used to compute the distance with respect to every sample. The *ordinal* property is when the ordering of the distance is taken into consideration. For example, mutual nearest neighbor is not ordinal since the nearest neighbors $\{a, b, c\}$ and $\{c, a, b\}$ are treated equally. The *batchable* property is a computational property that makes it feasible to compute in a reasonable time frame.

Vision-vision comparison In Figure 6.3, we evaluate Spearman's rank correlation among different metrics and hyperparameters over 78 vision models (details in Section 6.3.1). We find most metrics highly correlated with each other.

Cross-modal comparison We measure vision-language alignment using a range of alternative metrics. We visualize the corresponding alignment results in Figure 6.4 and Figure 6.5. Our findings indicate that alignment sensitivity not only depends on the metric used to compute it but also varies according to the specific tasks on which the vision models are trained.

Metric	Property				Description
	symmetric	global	ordinal	batchable	
CKA	✓	✓	✓	✓	Centered Kernel Alignment (CKA; Kornblith, Norouzi, Lee, et al. (2019)) measures the similarity of neural networks by comparing the alignment of their kernel induced by their feature spaces.
Unbiased CKA	✓	✓	✓	✓	Unbiased estimator of CKA that corrects for sample bias in HSIC Song, Smola, Gretton, et al. 2012.
SVCCA	✓	✓	✓	✓	Singular Value Canonical Correlation Analysis (SVCCA; Raghu, Gilmer, Yosinski, et al. (2017)) compares neural networks by decomposing their activities into singular vectors and measuring correlation.
Mutual k-NN	✓			✓	Measures the intersection over union (IoU) of nearest neighbors between two models.
CKNNA	✓	✓$*$	✓	✓	Modified CKA measure that computes the kernel alignment only for its nearest neighbors. See Appendix 6.1.
Cycle k-NN				✓	Measures whether the nearest neighbor in one domain also considers the original sample as its nearest neighbor in the other domain.
Edit k-NN	✓	✓$*$	✓		Computes the edit distance required to match the nearest neighbors between two datasets. The score is normalized by the maximum edit distance.
LCS k-NN	✓	✓$*$	✓		Calculates the longest common subsequence of nearest neighbors and is normalized by the sequence length.

Figure 6.2: Comparative analysis of neural network similarity metrics. ✓$*$ indicates the metric is global and still meaningful when the nearest neighbor k is set to maximum batch-size $k = |\mathcal{X}|$.

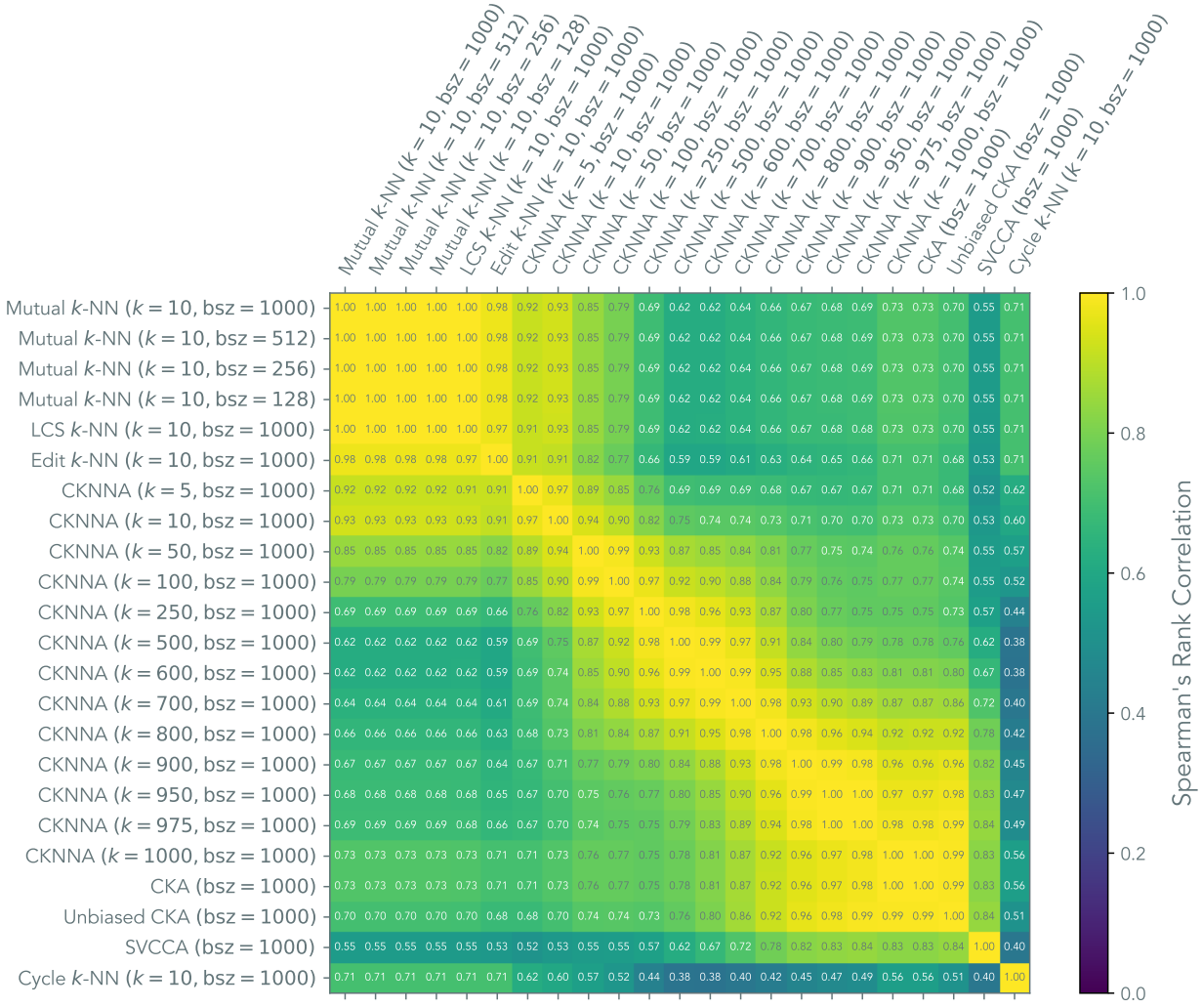
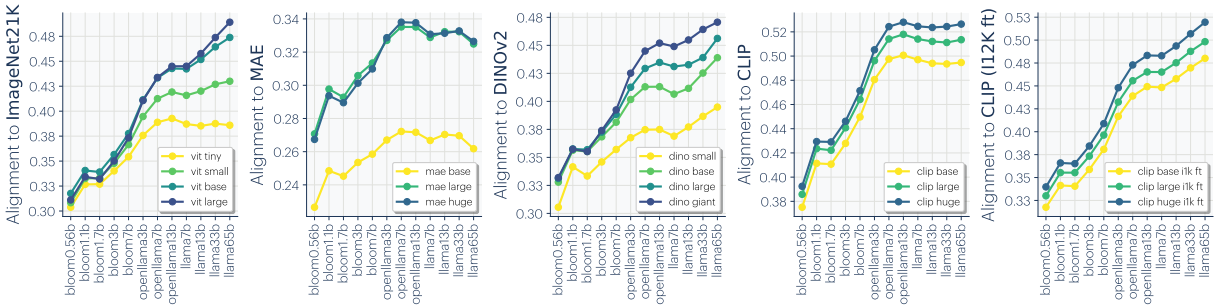
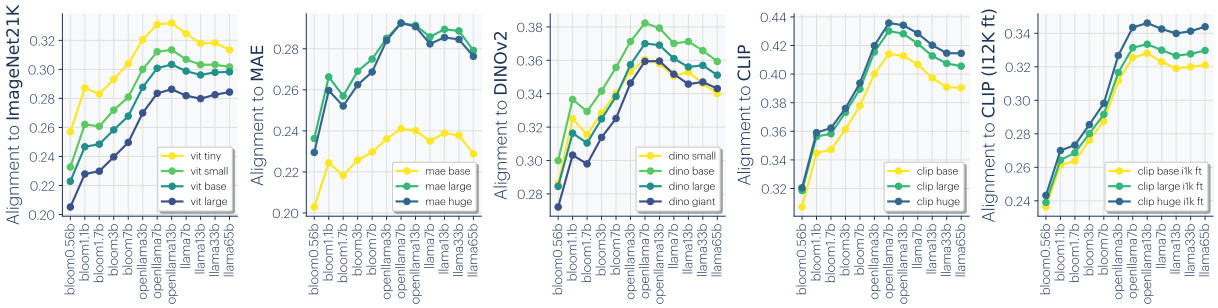


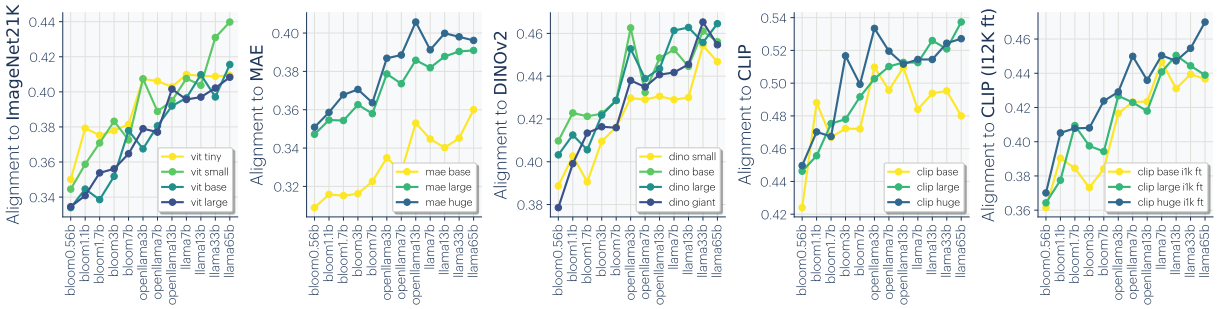
Figure 6.3: **Vision-vision alignment measured with various metrics.** Spearman's rank correlation among different metrics and batch sizes (bsz) when used to measure alignment among 78 vision models (see Section 6.3.1 for details of these models). All p-values are below 2.24×10^{-105}. Our vision-vision analysis in Figure 3.2 is based on the first metric (Mutual k-NN with $k = 10$ and bsz $= 1000$).



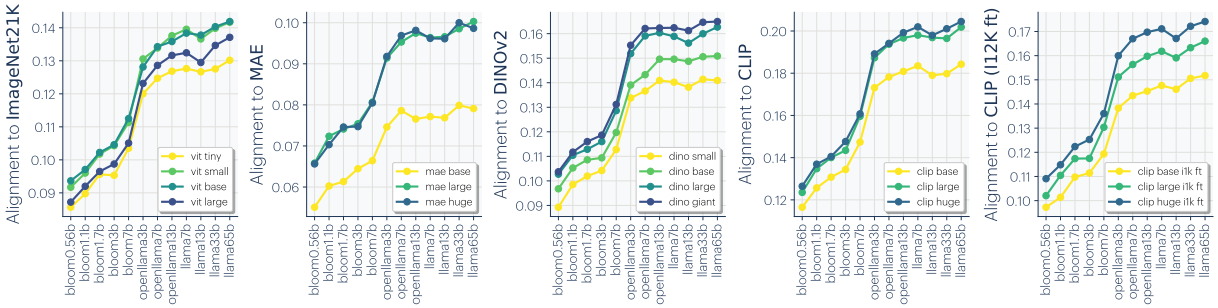
(a) CKA



(b) Unbiased CKA

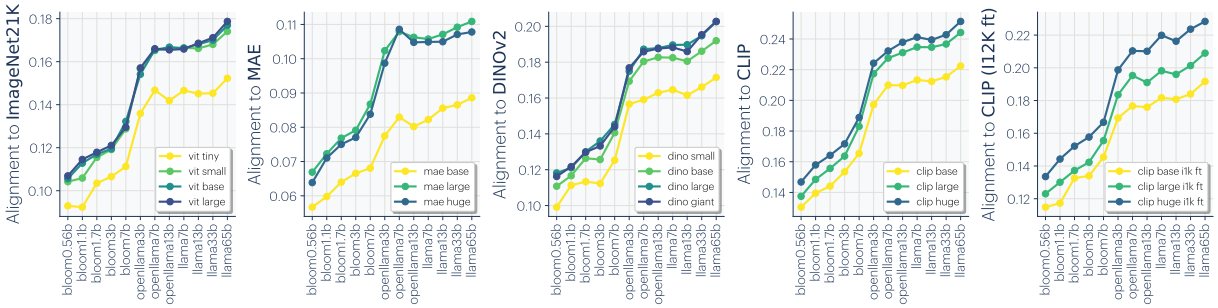


(c) SVCCA

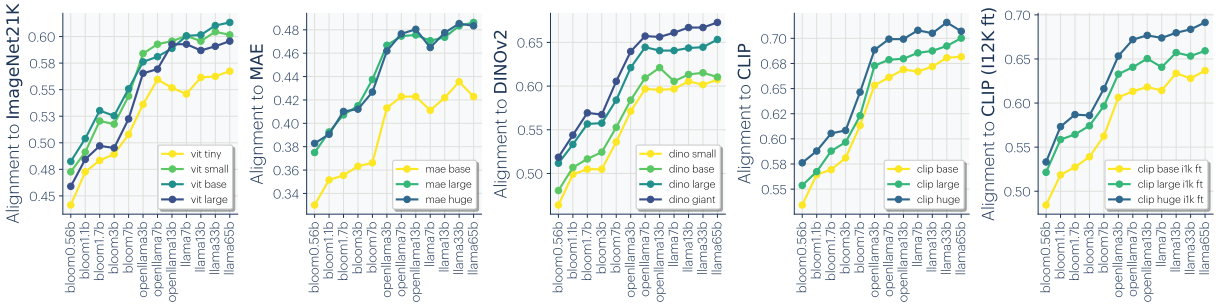


(d) Mutual k-NN ($k = 10$)

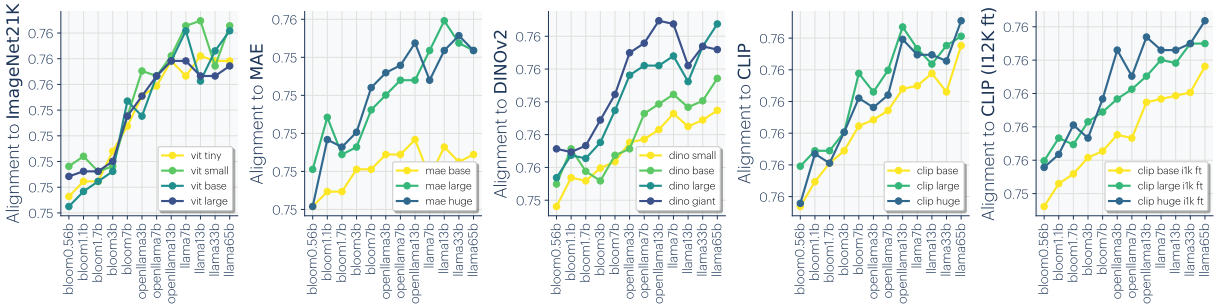
Figure 6.4: **Cross-modal alignment for various metrics**



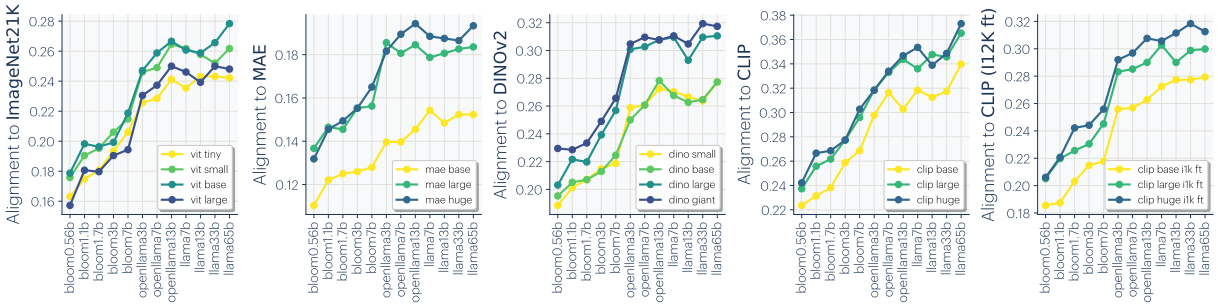
(a) CKNNA ($k = 10$)



(b) Cycle k-NN ($k = 10$)



(c) Edit-distance k-NN ($k = 10$)



(d) Longest-Common-Subsequence k-NN ($k = 10$)

Figure 6.5: **Cross-modal alignment measured with various metrics**

6.3 Experiments on Evaluating Alignment and Convergence

To demonstrate representational convergence, we take off-the-shelf models at multiple scales and multiple modalities and measure their representational alignment.

6.3.1 Vision-Vision Alignment and Representation Quality

We consider 78 vision models in total:

- 17 ViT models ranging from ViT-tiny to ViT-giant, trained on tasks including ImageNet-21k Dosovitskiy, Beyer, Kolesnikov, et al. 2020 classification, Masked Autoencoders He, Chen, Xie, et al. 2021, DINO Caron, Touvron, Misra, et al. 2021, and CLIP Radford, Kim, Hallacy, et al. 2021, including some finetuned on ImageNet-12k.
- 1 randomly initialized ResNet-50.
- 11 ResNet-50 models trained with contrastive learning on ImageNet-1k, Places-365 (Zhou, Lapedriza, Khosla, et al. 2017; Lopez-Cifuentes, Escudero-Vinolo, Bescos, et al. 2020), and 9 synthetic image datasets used in Baradad, Chen, Wulff, et al. (2022).
- 49 ResNet-18 models trained with Alignment and Uniformity contrastive loss (Wang and Isola 2020) on ImageNet-100, Places-365, and 47 realistic and synthetic image datasets from Baradad, Wulff, Wang, et al. (2021).

To test representation quality, we evaluate linear probing performance on all 19 VTAB classification tasks (Zhai, Puigcerver, Kolesnikov, et al. 2019), which is a standard multi-task transfer learning benchmark containing structured, specialized, and natural datasets covering diverse domains. To reduce compute requirements, we subsample training and validation datasets to have at most 10,000 samples. We consider a representation solves a task if its performance is $\geq 80\%$ of the best performance on that task across all 78 models.

To compute the alignment metric, we use $k = 10$ nearest neighbors over 1000 image representations computed on Places-365's validation dataset (Zhou, Lapedriza, Khosla, et al. 2017). This dataset is disjoint from VTAB datasets, although both contain natural images.

6.3.2 Cross-Modal Alignment

We compare the representation of an image in a vision model to the representation of a caption describing that image in a language model. The language model families we consider are BLOOM BigScience, Scao, Fan, et al. 2022, OpenLLaMA Geng and Liu 2023, and LLaMA Touvron, Martin, Stone, et al. 2023. For Figure 3.4, we included more recent model families such as OLMo Groeneveld, Beltagy, Walsh, et al. 2024, LLaMA3 Meta 2024, Gemma Team, Mesnard, Hardin, et al. 2024, and Mistral/Mixtral Jiang, Sablayrolles, Mensch, et al. 2023; Jiang, Sablayrolles, Roux, et al. 2024. These models were downloaded from Huggingface Wolf, Debut, Sanh, et al. 2019.

For vision models, we consider ViT models Dosovitskiy, Beyer, Kolesnikov, et al. 2020 of various sizes trained on various data and objectives. We mainly consider the popular vision models: classification on ImageNet-21K Russakovsky, Deng, Su, et al. 2015, MAE He, Chen, Xie, et al. 2021, DINOv2 Oquab, Darcet, Moutakanni, et al. 2023, CLIP Radford, Kim, Hallacy, et al. 2021, and CLIP finetuned on ImageNet-12K. These models were downloaded from PyTorch Image Models (TIMM; Wightman (2021)). This is a subset of the models used in vision-vision comparison.

To compute the alignment metric, we use $k = 10$ nearest neighbors over 1024 samples from WIT (Wikipedia-based Image Text)Srinivasan, Raman, Chen, et al. 2021. For the vision model, we use class token of each layer, and for the language model, we average pool each layer to a single token. Since it is not trivial to determine where the alignment might occur, we draw inspiration from BrainScoreSchrimpf, Kubilius, Hong, et al. 2018 and compute pairwise alignment scores, then take the maximum. One of these pairwise comparisons also includes concatenated features. We apply l_2 normalization to the features before measuring the distance. As transformer architectures have "emergent outliers" Dettmers, Lewis, Belkada, et al. 2022, we truncate the elements in the features that are above the 95-th percentile.

Simply taking the last token did not show any strong alignment signal. We also experimented with prompting the language model and taking the last token representation. The prompt we used was

```
An image with the caption '<caption>'.  This is an image of a <fill>
```

Using prompting showed similar trends to average pooling but had slightly lower alignment scores.

6.4 Color Cooccurrence Experiment

Here we describe the details of how we created the four color representations visualized in Figure 3.8, from left to right.

Perceptual representation from CIELAB color space We embed pixels taken from the CIFAR-10 image dataset (Krizhevsky, Hinton, et al. 2009; Torralba, Fergus, and Freeman 2008) based on the CIELAB color space, which is designed as a *perceptually uniform* space that changes numerical values correspond to similar perceived changes in color.

Three representations from cooccurrence in VISION and LANGUAGE For these three representations, we first obtain a dissimilarity matrix over colors (in different ways detailed below), then use multidimensional scaling (Shepard 1980) to find a 3-dimensional embedding in which Euclidean distance between the embeddings for A and B, z_A and z_B, best matches this dissimilarity matrix. We use 1,000 fits and take the best match. Afterward, we visually align it with the CIELAB space by finding the best rotation, translation, scaling, and flipping, by running the Kabsch-Umeyama algorithm (Kabsch 1976; Kabsch 1978; Umeyama 1991) twice, once on $\mathbf{z}$ and once on $-\mathbf{z}$, to account for flipping. The dissimilarity matrix we used in each case is described as following:

- **VISION: Pixel cooccurrence.** We collect color cooccurrence statistics from the CIFAR-10 dataset, and estimate a joint distribution $p(A, B)$ over 300,000 randomly sampled pixel colors A and B that occur within a radius of at most 4 pixels of one another. Colors are quantized on a grid in RGB space and represented as discrete variables, and $p(A, B)$ is modeled as a table of normalized counts, from which we compute the empirical pointwise mutual information matrix $K_{\mathsf{PMI}}(A, B)$. Quantization ensures that there is no bias from how color distances are represented in RGB space. Dissimilarity matrix is defined as $-K_{\mathsf{PMI}}(A, B) + c$, where $c = \max_{A,B} K_{\mathsf{PMI}}(A, B)$ is an offset to ensure non-negativity (similar to the constant in Section 3.4.2 and Proposition 6.6.1 that ensures neural networks can express K_{PMI}).
- **LANGUAGE.** We used an approach similar to Abdou, Kulmizev, Hershcovich, et al. (2021).
  - We take 20 pairs of (color, word) appeared in the dataset collected by Lindsey and Brown (2014), where 51 participants were asked to free name each of the 330 colors from the Munsell Color Chart. We filtered words that appeared less than 100 times, and computed each word's associate color by taking the centroid in CIELAB space. Our filtering process followed Abdou, Kulmizev, Hershcovich, et al. (2021) exactly, but resulted in 20 colors, a slightly different set than the 18 colors they claimed.
  - For each of the 20 color words `<col>`, we construct three sentences:

    ```
    The color <col>.
    This color is <col>.
    The color of this thing is <col>.
    ```

    and obtain the average sentence embedding from the language encoder, as the embedding for `<col>` (details below). We find this approach more effective than Abdou, Kulmizev, Hershcovich, et al. (2021), which uses object names that potentially have color biases, even though the objects may appear in multiple colors.
  - Unlike Abdou, Kulmizev, Hershcovich, et al. (2021), we did not perform linear regression from language embedding to CIELAB space, which distorts distances and easily overfits with only 20 samples. Instead, we used multidimensional scaling to best preserve distances, as described above.
  - **Masked language contrastive learning (SimCSE) embedding:** We used sentence embedding from the unsupervised SimCSE RoBERTa-L (Gao, Yao, and Chen 2021) to encode the above sentences into 1024-dimensional embeddings, and used the pairwise Euclidean distances among `<col>` embeddings as the dissimilarity matrix.
  - **Masked language predictive learning (RoBERTa) embedding:** We concatenated hidden states of the last four layers of RoBERTa-L (Liu, Ott, Goyal, et al. 2019), following (Devlin, Chang, Lee, et al. 2018). We averaged across token

dimensions, and obtained a 4096-dimensional embedding for each of the above sentences, and used the pairwise Euclidean distances among `<col>` embeddings as the dissimilarity matrix.

6.5 Caption Density Experiments

We use LLaMA3-8B-Instruct Meta 2024 to generate summary captions at various densities for images in the Densely Captioned Images dataset Urbanek, Bordes, Astolfi, et al. 2023 from the train split. Following Urbanek, Bordes, Astolfi, et al. (2023), we prompt the language model with the following instructions to generate captions at differing granularity:

```
system: You are given a full-text description of an image. You should
summarize it into about <num_words> words, being sure to include as much
salient visual information as possible given the <num_words> word
constraint, especially information from the start of the original
description. The new description should apply for the original image.
Respond with only the summary, in one line.

user: <original_caption>
```

6.6 Analysis of Contrastive Learners

6.6.1 Contrastive objectives learn pointwise mutual information

There are two widely used forms of contrastive objectives. We now discuss each form in details and show how they both are minimized by the pointwise mutual information (PMI) as stated in Equation (3.5). To simplify notation, we consider learning the bivariate model $g(x_a, x_b) \in \mathbb{R}$. In Section 3.4, such g is optimized within the family of $\{g = \langle f_X, f_X \rangle : f_X \in \mathcal{F}_X\}$.

Recall that our positive pairs are sampled from $(x, x_+) \sim P_{\mathsf{coor}}$, and that the negative pairs are sampled independently from its marginals which we denote as $(x, x_-) \overset{\text{i.i.d.}}{\sim} P$ where $P(x) = \sum_{x_+} P_{\mathsf{coor}}(x, x_+)$.

1. **The binary NCE loss (Gutmann and Hyvärinen 2010)** is defined with a certain prior over sampling positive vs. negative pairs. Let p_{pos} be the probability of sampling a positive pair. Then the loss is given by

$$\mathcal{L}_{\mathsf{binary\text{-}NCE}}(g) \triangleq p_{\mathsf{pos}} \cdot \mathbb{E}_{(x,x_+)\sim P_{\mathsf{coor}}} \left[-\log \sigma(g(x, x_+)) \right] \tag{6.12}$$

$$+ (1 - p_{\mathsf{pos}}) \cdot \mathbb{E}_{(x,x_-)\overset{\text{i.i.d.}}{\sim} P} \left[-\log \sigma(-g(x, x_-)) \right]. \tag{6.13}$$

The Bayes optimal solution is given by

$$g(x_a, x_b) = \log \frac{P(\texttt{pos} \mid x_a, x_b)}{1 - P(\texttt{pos} \mid x_a, x_b)} \tag{6.14}$$

$$= \log \frac{P(\texttt{pos}, x_a, x_b)}{P(\texttt{neg}, x_a, x_b)} \tag{6.15}$$

$$= \log \frac{p_{\mathsf{pos}} \cdot P_{\mathsf{coor}}(x_a, x_b)}{(1 - p_{\mathsf{pos}}) P(x_a) P(x_b)} \tag{6.16}$$

$$= \log \frac{P_{\mathsf{coor}}(x_a, x_b)}{P(x_a) P(x_b)} + \log \frac{p_{\mathsf{pos}}}{1 - p_{\mathsf{pos}}} \tag{6.17}$$

$$= K_{\mathsf{PMI}}(x_a, x_b) + c_X. \tag{6.18}$$

2. **The InfoNCE loss (Oord, Li, and Vinyals 2018)** is defined with randomly sampling one positive pair along with K negative ones. With some hyperparameter $\tau > 0$, the loss is given by

$$\mathcal{L}_{\mathsf{InfoNCE}}(g) \triangleq \mathbb{E}_{\substack{(x, x_+) \sim P_{\mathsf{coor}} \\ (x_-^{(1)}, x_-^{(2)}, \ldots, x_-^{(K)}) \overset{\text{i.i.d.}}{\sim} P}} \left[-\log \frac{e^{g(x, x_+)/\tau}}{e^{g(x, x_+)/\tau} + \sum_{i=1}^{K} e^{g(x, x_-^{(i)})/\tau}} \right]. \tag{6.19}$$

The Bayes optimal solution is given by

$$\frac{e^{g(x, x_+)/\tau}}{e^{g(x, x_+)/\tau} + \sum_{i=1}^{K} e^{g(x, x_-^{(i)})/\tau}} \tag{6.20}$$

$$= \frac{P_{\mathsf{coor}}(x_+ \mid x) \prod_j P(x_-^{(j)})}{P_{\mathsf{coor}}(x_+ \mid x) \prod_j P(x_-^{(j)}) + \sum_i P_{\mathsf{coor}}(x_-^{(i)} \mid x) P(x_+) \prod_{j \neq i} P(x_-^{(j)})} \tag{6.21}$$

$$= \frac{P_{\mathsf{coor}}(x_+ \mid x) / P(x_+)}{P_{\mathsf{coor}}(x_+ \mid x) / P(x_+) + \sum_i P_{\mathsf{coor}}(x_-^{(i)} \mid x) / P(x_-^{(i)})}. \tag{6.22}$$

For $\tau = 1$, this optima corresponds to g choices where

$$g(x_a, x_b) = \log \frac{P_{\mathsf{coor}}(x_b \mid x_a)}{P(x_b)} + c_X(x_a) \tag{6.23}$$

$$= K_{\mathsf{PMI}}(x_a, x_b) + c_X(x_a). \tag{6.24}$$

For the general $\tau \neq 1$ case, we have g (and corresponding f_X) recovers K_{PMI} up to an offset and a scale. Our main argument in Section 3.4 that f_X recovers K_{PMI} still holds.

6.6.2 Contrastive learners can represent K_{PMI} exactly under smoothness conditions

We want to express $K_{\mathsf{PMI}} + C$ using some representation function $f_X \colon \mathcal{X} \to \mathbb{R}^n$ so that

$$\langle f_X(x_a), f_X(x_b) \rangle = K_{\mathsf{PMI}}(x_a, x_b) + C, \qquad \text{for some } C. \tag{6.25}$$

For such an f_X to exist, an equivalent criterion is that $K_{\mathsf{PMI}} + C$ is positive semi-definite (PSD), as can be seen from eigendecomposition.

Proposition 6.6.1. *Suppose that the off-diagonal elements of K_{PMI} are bounded within $[\log \rho_{\mathsf{min}}, \log \rho_{\mathsf{min}} + \delta] \in (-\infty, 0]$. We have $K_{\mathsf{PMI}} + C$ is positive semi-definite (PSD) for some C if the joint distribution is sufficiently smooth:*

$$\frac{P_{\mathsf{coor}}(z_i \mid z_i)}{P_{\mathsf{coor}}(z_i)} \geq e^{N\delta} \rho_{\mathsf{min}}, \qquad \forall i. \tag{6.26}$$

Proof. Note that $K_{\mathsf{PMI}} + C$ still only has non-positive off-diagonal elements if

$$-C \geq \log \rho_{\mathsf{min}} + \delta. \tag{6.27}$$

For such C, it is diagonally dominant (and thus PSD) if,

$$\forall i, \qquad K_{\mathsf{PMI}}(z_i, z_i) + C \geq \sum_{j \neq i} |K_{\mathsf{PMI}}(z_i, z_j) + C| = -(N-1)C - \sum_{j \neq i} K_{\mathsf{PMI}}(z_i, z_j), \tag{6.28}$$

or equivalently,

$$\forall i, \qquad NC + \sum_j K_{\mathsf{PMI}}(z_i, z_j) \geq 0. \tag{6.29}$$

The following choice of C readily satisfies the above Equation (6.29):

$$C \triangleq -\min_i \frac{1}{N} \sum_j K_{\mathsf{PMI}}(z_i, z_j). \tag{6.30}$$

Therefore, it remains to show that Equation (6.27) is true. Note that

$$-C \triangleq \min_i \frac{1}{N} \sum_j K_{\mathsf{PMI}}(z_i, z_j) \geq \frac{N-1}{N} \log \rho_{\mathsf{min}} + \frac{1}{N} (\min_i K_{\mathsf{PMI}}(z_i, z_i)). \tag{6.31}$$

Therefore, it suffices to have

$$\log \rho_{\mathsf{min}} + \delta \leq \frac{N-1}{N} \log \rho_{\mathsf{min}} + \frac{1}{N} (\min_i K_{\mathsf{PMI}}(z_i, z_i)). \tag{6.32}$$

Rearranging terms gives the desired condition

$$\frac{P_{\mathsf{coor}}(z_i \mid z_i)}{P_{\mathsf{coor}}(z_i)} \geq e^{N\delta} \rho_{\mathsf{min}}, \qquad \forall i. \tag{6.33}$$

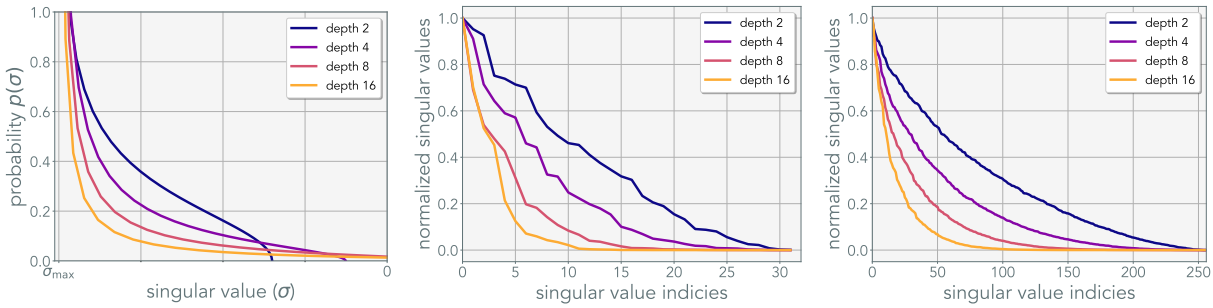
□

Remark 6.6.2. Proposition 6.6.1 is one example that a sufficiently smooth world or a sufficiently high sampling rate allows the PMI kernel K_{PMI} to be *exactly* represented as inner products of a learned feature space (up to a scale). The condition here can be satisfied, for example, if the off-diagonal terms decay linearly with respect to N and stay sufficiently close to each other. While the condition is somewhat strict, it captures the essence that smoothness and continuity allow easier learning. Nonetheless, we note that exact representation is not necessary for convergence, and thus this requirement can likely be relaxed. Please see Section 3.6 for discussions on practical settings.

Chapter 7

Chapter 3 Appendix

7.1 Random matrix theory in finite models

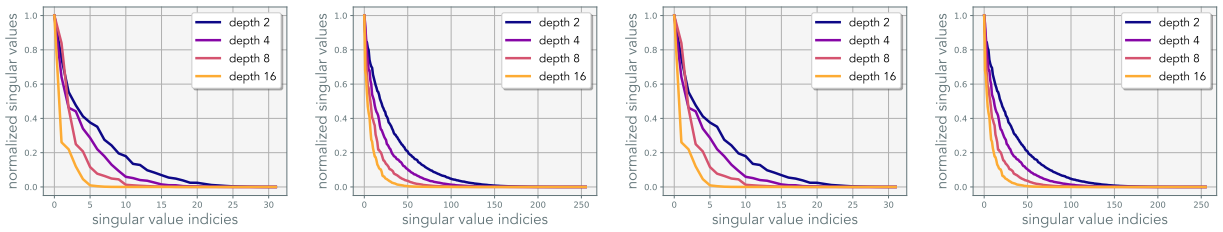


(a) Theoretical (b) Empirical $W \in \mathbb{R}^{32\times32}$ (c) Empirical $W \in \mathbb{R}^{256\times256}$

Figure 7.1: **Theoretical and empirical singular-value distributions:** We show that even on finite matrices, the singular-value distribution matches that of the theoretical distribution. This implies that deeper finite-width linear neural networks should have lower effective rank in practice. The theoretical distribution uses an unnormalized probability distribution.

Random matrix theory makes an infinitely large random matrix assumption (square or rectangular); one can think of them as infinitely wide neural networks. This infinitely large matrix assumption is used to derive a deterministic spectral distribution (singular-value distribution) of random matrices. In Figure 7.1, we show that the empirical spectral distribution closely follows that of the theoretical distribution derived in (Pennington, S. S. Schoenholz, and Ganguli 2017; Neuschel 2014). Even when using a very small weight matrix of size $W \in \mathbb{R}^{32\times32}$, and more so on larger weight matrices $W \in \mathbb{R}^{256\times256}$, the singular values are dominated by just a few values when increasing the number of layers.

In a similar light, we can also empirically observe the gram matrices' spectral distribution. As shown in Figure 7.2, we observed that gram matrices also exhibit almost the same trend predicted by random matrix theory. It is natural to assume that the theory has no practical meaning when the networks are trained, and the weight matrices are no longer random.



(a) Kernel rank $W \in \mathbb{R}^{32\times32}$ (b) Kernel rank $W \in \mathbb{R}^{32\times32}$ (c) Kernel rank $W \in \mathbb{R}^{256\times256}$ (d) Kernel rank $W \in \mathbb{R}^{256\times256}$

Figure 7.2: **Singular value distribution of Gram matrices:** Similar to the singular-value distribution of the weights, the singular-value distribution of gram matrices also becomes sharper, lower effective rank, with increased depth.

Hence we trained the models to convergence on least-squares objective and observed the spectral distribution to maintain its depth-wise separation as observed during initialization. These observations help reaffirm our conjecture and further motivate the potential usefulness of random matrix theory in understanding the role of over-parameterization in deep networks.

7.2 Expanding a non-linear network

A deep non-linear neural network with l layers is parameterized by a set of l weights $W = \{W_1, \ldots, W_l\}$. The output of the j-th layer is defined as $\phi_j = \psi(f_{W_j}(\phi_{j-1}))$, for some non-linear function ψ and input feature ϕ_{j-1}. The initial feature map is the input $\phi_0 = x$, and the output is the final feature map $y = \phi_l$. We can expand a model by depth d by expanding all linear layers, i.e. redefining $f_{W_j} \rightarrow f_{W_j^d} \circ \cdots \circ f_{W_j^1} \;\forall\; j \in \{1, ..., l\}$. We illustrate this in Figure 7.3. We describe this operation for fully connected and convolutional layers.

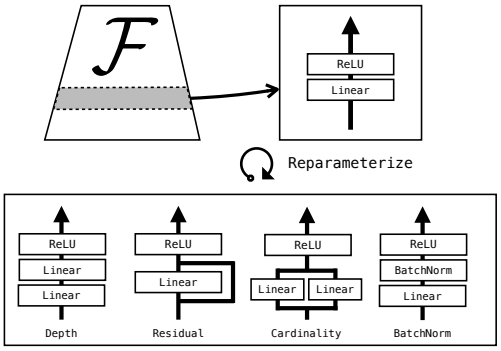


Figure 7.3: **Linear reparameterization**: For a model $\mathcal{F}$, we can reparameterize any linear layer to another functionally equivalent layer (shown in the box below). In this work we mainly explore reparameterization of depth. Batch-norm and any other running-statistics driven normalization layers are linear only at test time.

Fully-connected layer A fully-connected layer is parameterized by weight $W \in \mathbb{R}^{m\times n}$. One can over-parameterize W as a series of linear operators defined as $\prod_{i=1}^{d} W_i$. For example, when $d = 2$, $W \rightarrow W_2W_1$, where $W_2 \in \mathbb{R}^{m\times h}$ and $W_1 \in \mathbb{R}^{h\times n}$ for some hidden dimension h. The variable h is referred to as the width of the expansion and can be arbitrarily chosen. In our experiments, we choose $h = n$ unless stated otherwise. Note that $h < \min(m, n)$ would result in a rank bottleneck and explicitly reduce the underlying rank of the network.

Convolutional layer A convolutional layer is parameterized by weight $W \in \mathbb{R}^{m \times n \times k \times k}$, where m and n are the output and input channels, respectively, and k is the dimensionality of the convolution kernel. For convenience, we over-parameterize by adding 1×1 convolution operations. $W_d * W_{d-1} * \cdots * W_1$, where $W_d \in \mathbb{R}^{m \times h \times 1 \times 1}$, $W_{d-1}, ..., W_2 \in \mathbb{R}^{h \times h \times 1 \times 1}$ and $W_1 \in \mathbb{R}^{h \times n \times k \times k}$. Analogous to the fully-connected layer, we choose $h = n$ to avoid rank bottleneck.

The work by Golubeva, Neyshabur, and Gur-Ari (2021) explores the impact of width h. Similar to their findings, we observed using the larger expansion width to slightly improve performance. We use $h = 2n$ for our ImageNet experiments.

7.3 Comparisons of rank measures and kernel distance functions

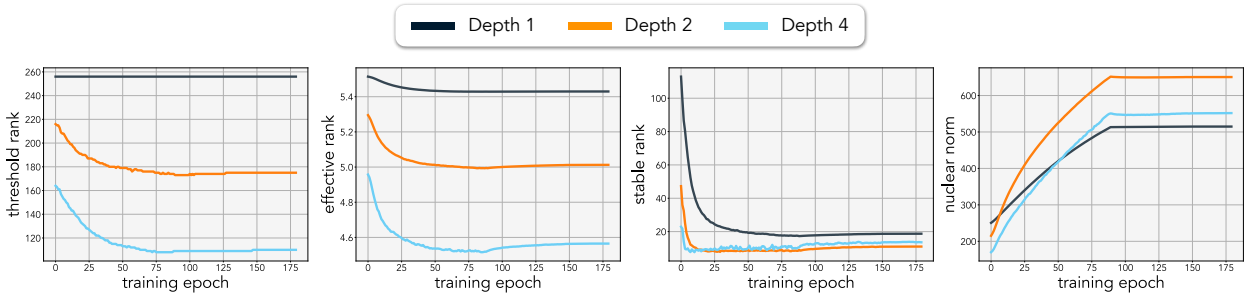


Figure 7.4: **Comparing rank-measures:** Comparison between various pseudo-metrics of rank when varying the number of layers. The threshold is set to $\tau = 0.01$ for threshold rank.

The rank of a matrix – which defines the number of independent basis – in practice can often be a sub-optimal measure. For deep learning, fluctuations in stochastic gradient descent and numerical imprecision can easily introduce noise that causes a matrix to be full-rank. In addition, simply counting the number of non-zero singular values may not indicate what we care about in practice: the relative impact of the i-th basis compared to the j-th basis. In a typical image classification setup, we observed that the norm of the matrix often increases during training. This is highlighted by the nuclear norm in Figure 7.4. Coupled with numerical imprecisions, we found that the weights of the matrix are often always full rank.

A rank measure closest to the true definition of rank would be thresholded-rank, where the smallest singular values are thresholded after normalization (re-weighting the singular values based on relative contribution). However, thresholded rank is very sensitive to the threshold value one chooses (shown below); hence we used effective rank to avoid this issue.

Definition 7.3.1 (Effective rank)**.** (Roy and Vetterli 2007)

For any matrix $A \in \mathbb{R}^{m \times n}$, the effective rank ρ is defined as the Shannon entropy of the normalized singular values:

$$\rho(A) = -\sum_{i=1}^{\min(n,m)} \bar{\sigma}_i \log(\bar{\sigma}_i),$$

where $\bar{\sigma}_i = \sigma_i / \sum_j \sigma_j$ are the normalized singular values such that $\sum_i \bar{\sigma}_i = 1$. It follows that $\rho(A) \leq \text{rank}(A)$. This measure is also known as the spectral entropy.

The effective rank has been previously used as a surrogate measure for measuring the rank of neural network weights ((Arora, Cohen, Hu, et al. 2019)). We now state other various metrics that have been used as a pseudo-measure of matrix rank. One obvious alternative is to use the original definition of rank after normalization:

Definition 7.3.2 (Threshold rank)**.** For any matrix $A \in \mathbb{R}^{m \times n}$, the threshold rank τ-*Rank* is the count of non-small singular values after normalization:

$$\tau\text{-}\mathsf{Rank}(A) = \sum_{i=1}^{\min(n,m)} \mathbb{1}[\bar{\sigma}_i \geq \tau],$$

where $\mathbb{1}$ is the indicator function, and $\tau \in [0, 1)$ is the threshold value. $\bar{\sigma}_i$ are the normalized singular values defined above.

It is worth noting that not normalizing the singular values results in the numerical definition of rank. As stated before, the threshold rank depends largely on the threshold value and therefore a drastically different scalar representation of rank can emerge. Potentially, a better usage of threshold rank is to measure the AUC when varying the threshold.

Related to the definition of the threshold rank, stable rank operates on the normalized squared-singular values:

Definition 7.3.3 (Stable rank)**.** (Vershynin 2018)

For any matrix, $A \in \mathbb{R}^{m \times n}$, the stable rank is defined as:

$$\mathsf{SRank}(A) = \frac{\|A\|_F^2}{\|A\|_2} = \frac{\sum \sigma_i^2}{\sigma_{max}^2},$$

Where σ_i are the singular values of A.

Stable-rank provides the benefit of being efficient to approximate via the power iteration (Mises and Pollaczek-Geiringer 1929). In general, stable-rank is a good proxy for measuring the rank of the matrix and has been used in prior works such as (Nichani, Radhakrishnan, and Uhler 2020). This is not necessarily true when the singular values have a long tail distribution, which under-emphasizes the small singular values un-proportionately due to the squared-operator. We observed that the largest singular values often get over exaggerated in neural networks and hence we often found that SRrank converges to values close to 1, making insightful observations impractical.

Lastly, the nuclear norm has been considered as the de facto measure of rank for the task of matrix factorization/completion, with low nuclear-norm indicating that the matrix is low-rank:

Definition 7.3.4 (Nuclear norm)**.** For any matrix $A \in \mathbb{R}^{m \times n}$, the nuclear norm operator is defined as:

$$\|A\|_* = \text{tr}(\sqrt{AA^T}) = \sum_{i}^{\min(n,m)} \sigma_i(A)$$

Where σ_i are the singular values of A.

Nuclear norm, however, has obvious flaws of being an un-normalized measure. The nuclear norm is dictated by the magnitude of the singular values and not the ratios. Therefore, the nuclear norm can be made arbitrarily large or small without changing the output distribution.

The comparisons of these metrics are illustrated in Figure 7.4 where effective rank has the closest behavior to that of the thresholded rank. The metrics are computed on the end-to-end weights throughout the training. We use linear over-parameterized models with various depths on least-squares.

In Figure 7.5, we repeat our least-squares experiments from our main paper using thresholded rank with various threshold values $\tau = \{0.001, 0.005, 0.01\}$. We show that the effective rank indeed correlates well with the thresholded rank. As stated above, we observe that the rank drastically changes depending on the threshold value. We also run the same experiment on varying task-ranks of 30, 16, and 4. Although all models span the same set of functions (same effective weight dimensionality), the resulting generalization performance differs depending on the depth of the model. In a high task-rank setting, the generalization error increases with depth, while generalization error decreases with depth in a low task-rank setting. This indicates the parameterization of the model determines the hypothesis space the model explores during training, which aligns with our conjecture and our observations. This is further highlighted in medium and low task-rank settings, where all models reach zero-training error, yet the test-loss differs.

Kernel distance functions In our work we used cosine kernels to construct the Gram matrices. Cosine kernels are normalized linear kernels and we found it to produce cleaner results. Cosine kernels has been commonly used as distance function to measure similarity between features (Zhang, Isola, Efros, et al. 2018). We further show in Figure 7.6 that Gram matrices constructed with kernel distance functions such as linear kernels and correlation kernels also exhibit the low-rank simplicity bias.

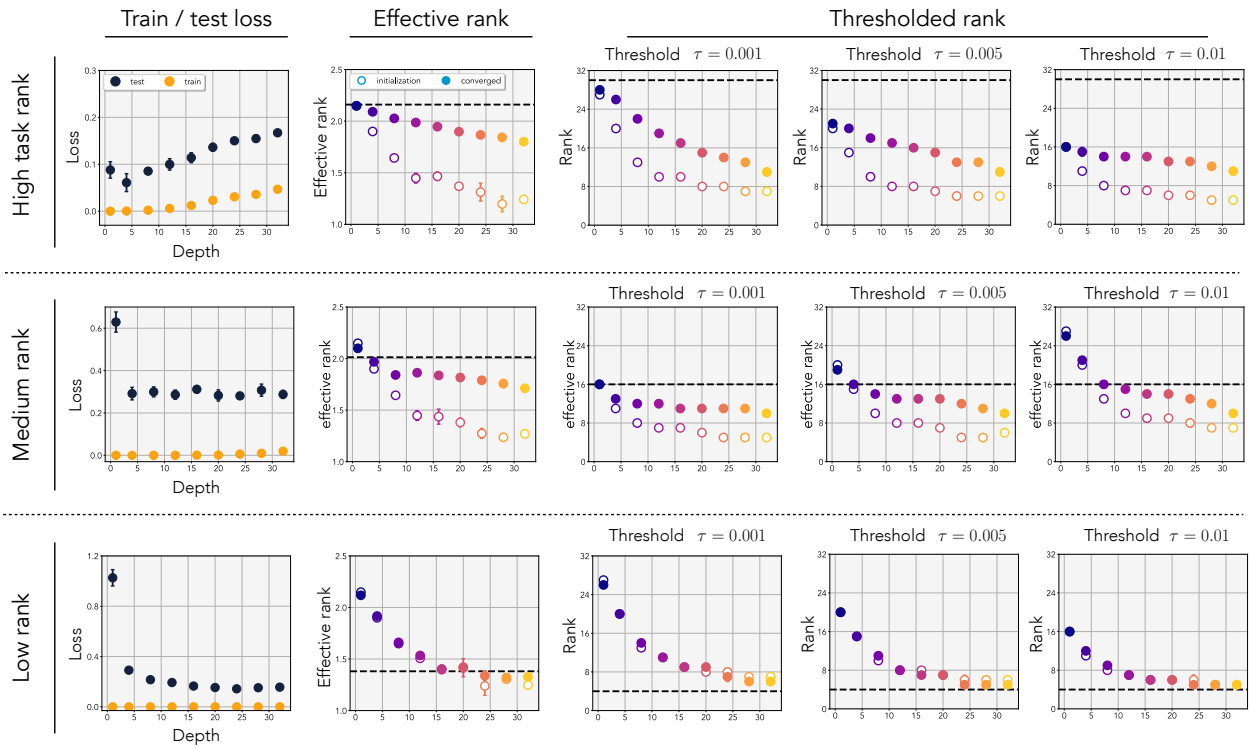


Figure 7.5: **Least-sqaures ablation**: Least-squares experiment using both effective-rank and thresholded rank measure. We run the experiments on various task-ranks $30, 16, 4$. For thresholded rank, we use various threshold values of $\tau = \{0.001, 0.005, 0.01\}$ and show that it correlates well with effective rank. The thresholded rank has a downside of being sensitive to the threshold values, and one has to subjectively tune the suitable threshold, making it a suboptimal choice. The figure shows that depending on the rank of the task, the generalization performance depends on the depth. When the task rank is high, shallower models perform better, and when the task rank is low, deeper models perform better. This aligns with our observation that the model parameterization biases the hypothesis search space in neural networks even if the models are effectively the same and span the same set of functions.

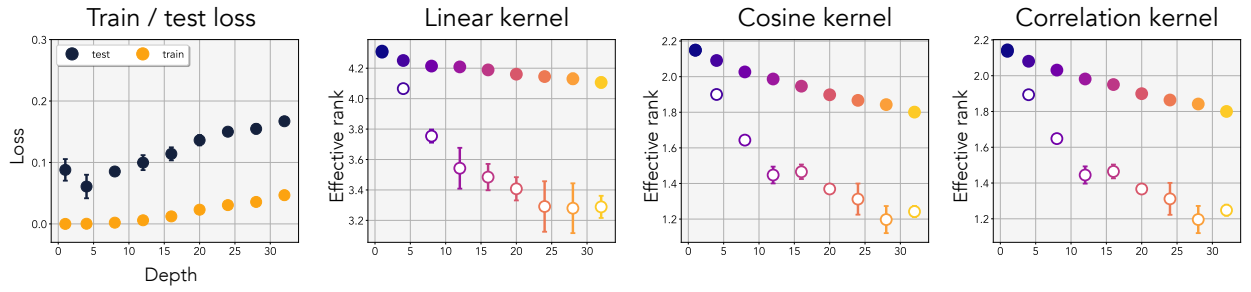


Figure 7.6: **Kernel ablation**: We ablate our least-squares experiments by using various kernel distance functions. Cosine kernels are normalized version of linear kernels, and pearseon-correlation kernels are another way of normalizing linear kernels. We can see that all kernels show the same behavior.

7.4 Singular value dynamics of weights

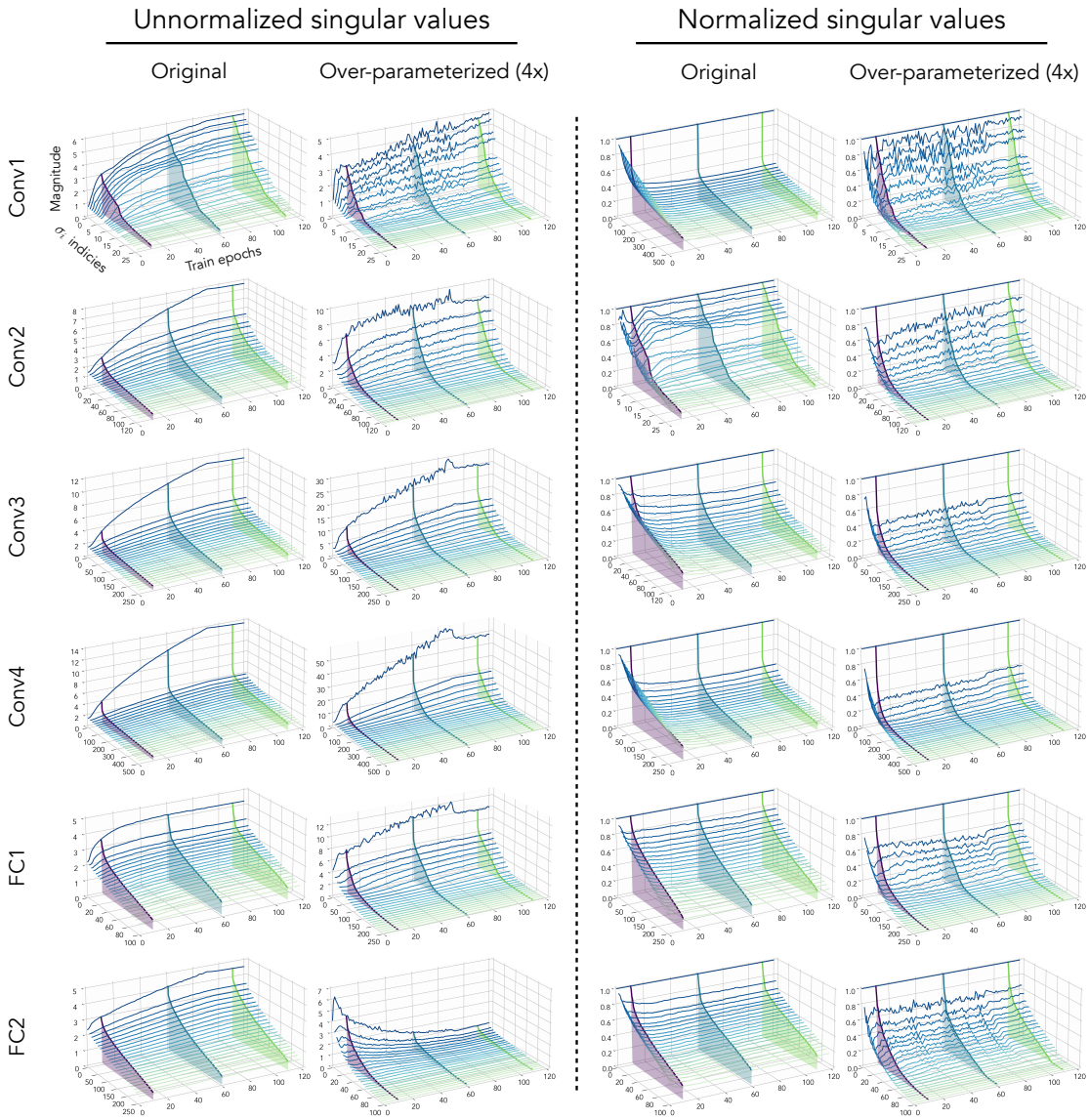


Figure 7.7: **Dynamics per layer**: The singular values of the individual weights during `CIFAR100` training. On the left we have the unnormalized singular values, and on the right the distributions are scaled by the largest singular values. We uniformly subsample 24 singular values for the visualization. The cross sections are provided to help visualize the distribution at that specific epoch. The individual lines track the singular values σ_i over time.

In Figure 7.7, we visualize the singular values of the individual weights when training on `CIFAR100` image classification for the first 120 epochs. The cross-sections indicate the singular value distribution at that specific epochs. For the over-parameterized model, the effective rank is computed on the effective weight. On the left, we plot the unnormalized singular values and observe that the norm of the singular values increases throughout training for all layers except for the last classification layer in the over-parameterized model. When we

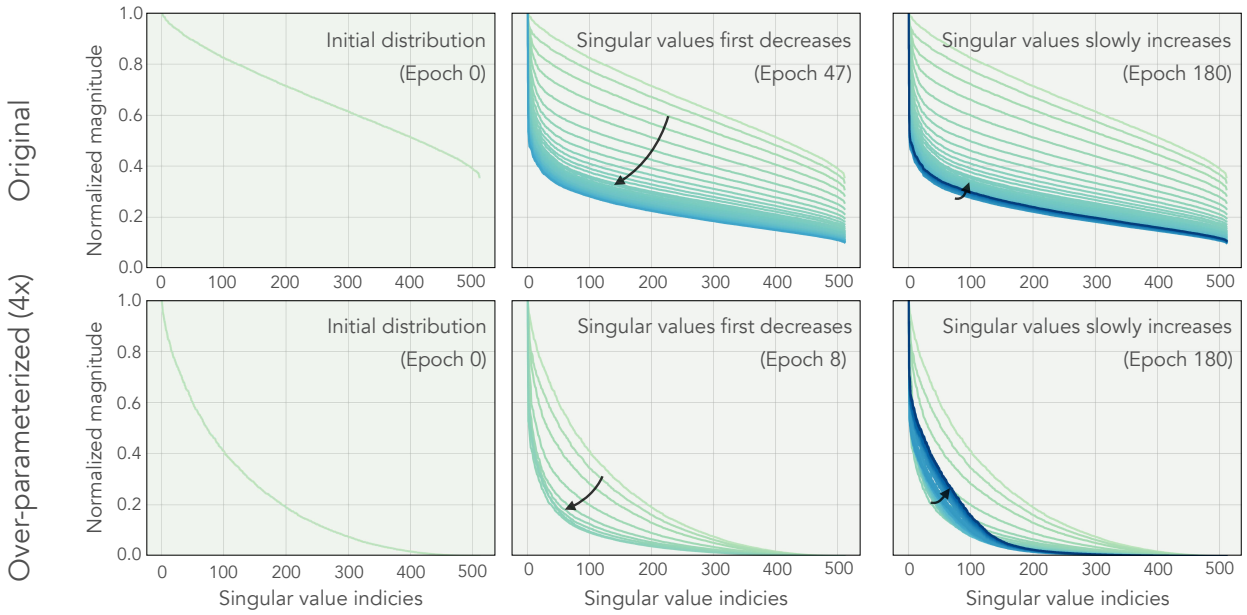


Figure 7.8: **Dynamics overlay**: We overlay the singular values of the `Conv4` weights. We observed that the effective rank first rapidly decreases early on in the training and then bounces back up slowly throughout the rest.

normalized the distribution by the largest singular value σ_0 (right), we observed that the distribution becomes sharper early in training but does not change much throughout.

To get a better sense of how the distribution evolves over time by overlaying the distribution on top of each other. In Figure 7.8, we overlay the distribution on top of each other for `Conv4` weights and observed that the effective-rank first decays rapidly and then slightly increases throughout the rest of the training. This dynamical behavior, to our knowledge, is not explained in prior theoretical works and could highlight the dissonance between the assumptions made in theory do not fully describe behaviors observed in practice.

7.5 Training details and model architecture

For Figure 2.1, we trained a ReLU network with input, output, and the hidden dimension of 64; the larger the width, the more pronounced the effects seemed to be. We chose 64 due to the run time of these models. We train the model using SGD with a momentum of 0.9, and we do not use weight decay. We observed that very deep networks become very sensitive to the learning rate. Therefore, we tuned the learning rate per model. For each model we trained using the learning rates $[1.0, 0.5, 0.2, 0.1, 0.05, 0.02, 0.01, 0.005, 0.002, 0.001]$ and chose the best performing learning rate. A heuristics we found somewhat helpful is setting the learning rate as $\eta \propto \frac{1}{\sqrt{d}}$ for some depth d. The weights are initialized using normal distribution and linearly swept through the scale of the variance. We found the gain of $\sqrt{2}$, the default gain of Kaiming initialization (He, Zhang, Ren, et al. 2016), to work the best. We tried 5 different seeds for each task-rank and also 5 different initialization seed for the neural

network and observed a consistent result. All models are trained for 24000 epochs as we observed deeper models take a long time to converge. For shallower models, it is sufficient to train them for roughly 1000 epochs. We also experimented with learning rate schedulers but only helped a little. For all models, we step the learning rate by a factor of 10 at epoch 18000. For all experiments $\text{rank}(W^*) = \{1, 4, 16, 32, 64\}$, we use total of 128 training samples. Using a different number of training samples results in similar observations. We experimented with both SGD and GD and observed the same phenomena. For SGD, we used a mini-batch size of 32. **When the rank of the underlying function is high, we found that it required significantly more fine-grained tuning of the hyper-parameters.**

All models for image classification are trained using `PyTorch` (Paszke, Gross, Massa, et al. 2019) with `RTX 2080Ti` GPUs. We use stochastic gradient descent with a momentum of 0.9. For CIFAR experiments, the initial learning rate is individually tuned (0.02 for most cases), and we train the model for 180 epochs. We use a step learning rate scheduler at epoch 90 and 150, decreasing the learning rate by a factor of 10 each step. For all the models, we use random-horizontal-flip and random-resize-crop for data augmentation.

The training details for ImageNet can be found in https://github.com/pytorch/examples/blob/master/imagenet. When linearly over-parameterizing our models, we bound the variance of the weights using Kaiming initialization (He, Zhang, Ren, et al. 2016), a scaled Normal distribution. This allows us to have the same output variance, regardless of the number of layers we over-parameterize our models by. We found this to be critical for stabilizing our training. We also found it important to re-tune the weight decay for larger models on ImageNet. The architecture used for the `CIFAR` experiments is:

CIFAR architecture
RGB image $y \in \mathbb{R}^{32\times32\times3}$
Convolution $3 \to 64$, MaxPool, ReLU
Convolution $64 \to 128$, MaxPool, ReLU
Convolution $128 \to 256$, MaxPool, ReLU
Convolution $256 \to 512$, ReLU
GlobalMaxPool
Fully-Connected $512 \to 256$, ReLU
Fully-Connected $256 \to$ num classes

We tuned the learning rate per model as deeper models (8x expansion or more) become sensitive to the initial learning rate. This was critical for the least-squares experiments but not so much for `CIFAR` and `ImageNet` experiments (since we used up to 8x expansion). The one hyper-parameter that we found that needed tuning was the weight decay in `ImageNet`

classification. A typical 2x or 4x expansion does not require much tuning at all. The learning rate scheduler was originally tuned to the baseline and was held fixed. The learning rate decay for baselines with explicit regularizers was tuned.

For least-squares experiments, we were unable to achieve zero-training error for very deep networks using various common optimization tricks. We argue that the parametric bias of depth is the reason why these models are unable to overfit to high-rank data. While it is certainly possible that an optimizer or optimization setting would allow us to reach zero-training error, we were unable to find such a setting by sweeping across hyper-parameters and common optimization techniques. Given an SGD optimizer, we tuned learning rates ($\{0.1, 0.05, 0.01, 0.005, 0.001\}$), momentum (($\{0.1, 0.5, 0.9, 0.99\}$), learning rate schedulers (none, step, decay-on-plateau, cosine). We found the best set of hyperparameters that minimizes the training loss is with momentum set to 0.9 and using decay-on-plateau scheduler. For an over-parameterized linear network of depth 16, and underlying task rank set to 24, we show that even with the best set of hyper-parameters, the training loss cannot be perfectly minimized:

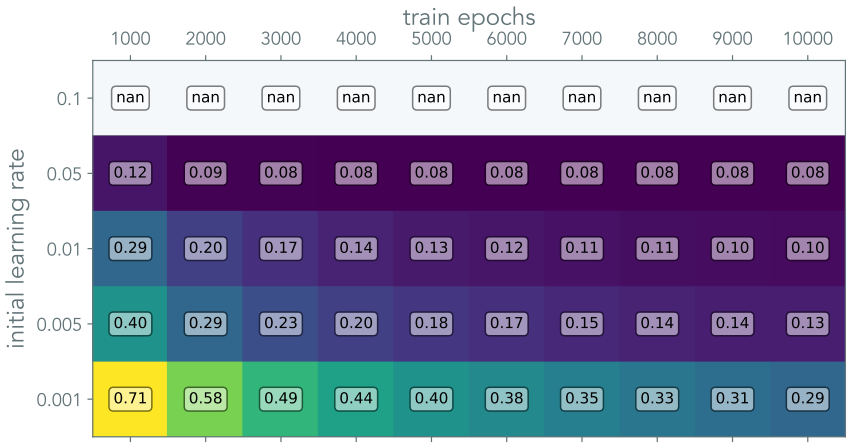


Figure 7.9: **Training error vs learning-rate:** Training error with varying learning rates for least-squares trained on a linear network with a depth of 24.

7.6 Differential effective rank

To analyze the effective rank as a function of the number of layers, we define a differential variant of the effective rank. This formulation allows us to use the fact that the eigen/singular-value spectrum assumes a probability distribution in the asymptotic case.

Definition 7.6.1 (Differential effective rank)**.** For any matrix $A \in \mathbb{R}^{m \times n}$ as $\min(m, n) \to \infty$ the singular values assume a probability distribution $p(\sigma)$. Then, we define the differential effective rank ρ as:

$$\rho(A) = -\int_0^{\sigma_{\max}} \frac{\sigma}{c} \log(\frac{\sigma}{c}) p(\sigma) d\sigma \tag{7.1}$$

where $p(\sigma)$ is the singular value density function and $c = \int_0^{\sigma_{\max}} \sigma p(\sigma) d\sigma$ is the normalization constant.

7.7 Proof of Theorem 1

To prove Theorem 2.4.1, we leverage the findings from random matrix theory, where the singular values assume a probability density function. Specifically, we use the density function corresponding to the singular values of the matrix W composed of the product of L individual matrices $W = W_L, \dots W_1$, where the components of the matrices W_1 to W_L are drawn i.i.d from a Gaussian. Characterizing such density function is, in general intractable, or otherwise very difficult. However, in the asymptotic case where $dim(W) \to \infty$ and W is square, the density function admits the following concise closed-form (Eq. 13 of Pennington, S. S. Schoenholz, and Ganguli (2017) derived from Neuschel (2014)):

$$p(\sigma(\phi)) = \frac{2}{\pi} \sqrt{\frac{\sin^3(\phi) \sin^{L-2}(L\phi)}{\sin^{L-1}((L+1)\phi)}} \quad \sigma(\phi) = \sqrt{\frac{\sin^{L+1}((L+1)\phi)}{\sin(\phi) \sin^L(L\phi)}}\,, \tag{7.2}$$

where σ denotes singular values (parameterized by $\phi \in [0, \frac{\pi}{L+1}]$) and p denotes the probability density function of σ for $\sigma \in [0, \sigma_{\max}]$, and $\sigma_{\max}^2 = L^{-L}(L+1)^{L+1}$. The parametric probability density function spans the whole singular value spectrum when sweeping the variable ϕ.

We are interested in computing the effective rank of W. Using the above density function, we can write it in the form:

$$\rho(W) = -\int_0^{\sigma_{\max}} \frac{\sigma}{c} \log(\frac{\sigma}{c}) p(\sigma)\, d\sigma\,, \tag{7.3}$$

We now write this integral in terms of ϕ as the integration variable, such that we can leverage the density function in Eqn. 7.2. Using the change of variable, we have:

$$\rho(W; L) \;=\; -\int_0^{\frac{\pi}{L+1}} \frac{\sigma(\phi)}{c} \log(\frac{\sigma(\phi)}{c}) \big(- p(\sigma(\phi))\sigma'(\phi) \big)\, d\phi\,, \tag{7.4}$$

where $\sigma'(\phi) = \frac{d}{d\phi}\sigma(\phi)$. Note that the integral limits $[0, \sigma_{\max}]$ on σ respectively translate[1] into $[\frac{\pi}{L+1}, 0]$ on ϕ, where,

$$-p(\sigma(\phi))\sigma'(\phi) = \frac{1}{2\pi} \Big(1 + L + L^2 - L(L+1)\cos(2\phi) - (L+1)\cos(2L\phi) + L\cos(2(1+L)\phi) \Big) \csc^2(L\phi)\,.$$

[1] note that the direction of integration needs to flip (by multiplying by -1) to account for flip of the upper and lower limits.

In the following, we treat L as a continuous variable, and show that $\rho(W;L)$ is decreasing in L. This is sufficient for proving $\rho(W;L)$ results in a decreasing sequence at integer values of L.

As $\rho(W;L)$ is differentiable in L, $\rho(W;L)$ decreases in L if and only if $\frac{d\rho}{dL} < 0$. Since integration and differentiation are w.r.t. different variables, they commute; we can first compute the derivative of the integrand w.r.t. L and then integrate w.r.t. ϕ and show that the result is negative.

With abuse of notation, we rewrite Eqn. 7.4 by making the dependency of functions on L explicit.

$$\rho(W;L) \;=\; \int_0^{\frac{\pi}{L+1}} \frac{\sigma(\phi,L)}{c(L)} \log(\frac{\sigma(\phi,L)}{c(L)}) p(\sigma(\phi,L))\sigma'(\phi,L)\, d\phi\,, \tag{7.5}$$

where $\sigma'(\,.\,,\,.\,)$ denotes partial derivative of $\sigma(\,.\,,\,.\,)$ w.r.t. its first argument.

We now proceed with differentiating ρ w.r.t. L. Notice that, besides the integrand, the integral limit depends on L as well. Thus can be handled using Leibniz integral rule for differentiation, which yields,

$$\frac{\partial \rho}{\partial L} = \quad \Big(\frac{\sigma(\phi,L)}{c(L)} \log(\frac{\sigma(\phi,L)}{c(L)}) p(\sigma(\phi,L))\sigma'(\phi,L)\Big)_{\phi\to\frac{\pi}{L+1}} \Big(\frac{\partial}{\partial L}\frac{\pi}{L+1}\Big) \tag{7.6}$$

$$+ \quad \int_0^{\frac{\pi}{L+1}} \frac{\partial}{\partial L}\Big(\frac{\sigma(\phi,L)}{c(L)} \log(\frac{\sigma(\phi,L)}{c(L)}) p(\sigma(\phi,L))\sigma'(\phi,L)\Big)\, d\phi \tag{7.7}$$

It is easy to verify that,

$$\lim_{\phi\to\frac{\pi}{L+1}} \frac{\sigma(\phi,L)}{c(L)} \log\left(\frac{\sigma(\phi,L)}{c(L)}\right) = 0$$

$$\lim_{\phi\to\frac{\pi}{L+1}} p(\sigma(\phi,L))\sigma'(\phi,L) = \frac{(L+1)\Big(L\cos\left(\frac{2\pi}{1+L}\right) + \cos\left(\frac{2L\pi}{1+L}\right) - 1 - L\Big)\csc^2\left(\frac{L\pi}{1+L}\right)}{2\pi} \tag{7.8}$$

Consequently,

$$\Big(\frac{\sigma(\phi,L)}{c(L)} \log(\frac{\sigma(\phi,L)}{c(L)}) p(\sigma(\phi,L))\sigma'(\phi,L)\Big)_{\phi\to\frac{\pi}{L+1}} \;=\; 0\,. \tag{7.9}$$

This allows us to drop the first term in $\frac{\partial\rho}{\partial L}$ to express it more compactly as,

$$\frac{\partial \rho}{\partial L} \;=\; \int_0^{\frac{\pi}{L+1}} \frac{\partial}{\partial L}\Big(\frac{\sigma(\phi,L)}{c(L)} \log(\frac{\sigma(\phi,L)}{c(L)}) p(\sigma(\phi,L))\sigma'(\phi,L)\Big)\, d\phi\,. \tag{7.10}$$

It is messy but straightforward to compute $\frac{\partial}{\partial L}\Big(\frac{\sigma(\phi,L)}{c(L)} \log(\frac{\sigma(\phi,L)}{c(L)}) p(\sigma(\phi,L))\sigma'(\phi,L)\Big)$. Integrating that w.r.t. ϕ from 0 to $\frac{\pi}{L+1}$ leads to a negative expression, thus $\frac{\partial\rho}{\partial L} < 0$.

The proof here considers the asymptotic case when $\dim(W) \to \infty$. This limit case allowed us to use the probability distribution of the singular values. Although we do not provide proof for the finite case, our results demonstrate that it holds empirically in practice (see Figure 2.2).

7.8 Extension to residual connections

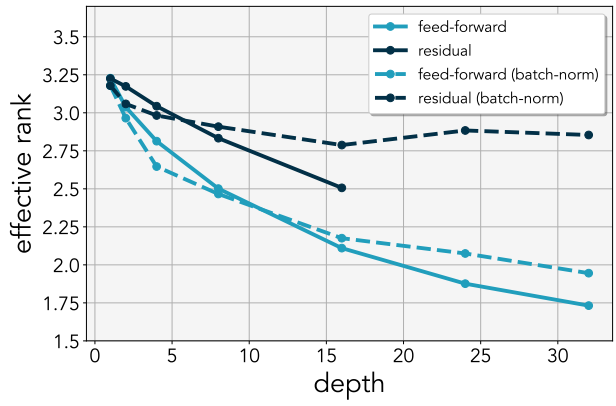


Figure 7.10: **Residual connections:** The effective rank of linear models trained with and without residual connection on a low-rank least-squares problem. Contrary to feed-forward networks, residual networks maintain the effective rank of the weights even when adding more layers. Residual networks without batch-normalization suffer from unstable output variance after 16 layers.

This work concentrates our analysis on depth and its role in both linear and non-linear networks. Yet, the ingredients that make up what we know as state-of-the-art models today are more than just depth. From cardinality (Xie, Girshick, Dollár, et al. 2017) to normalization (Ioffe and Szegedy 2015) and residual connections (He, Zhang, Ren, et al. 2016), numerous facets of parameterization have become a fundamental recipe for a successful model (see Figure 7.3). Of these, residual connections have the closest relevance to our work.

What is it about residual connections that allow the model to scale arbitrarily in depth? while vanilla feed-forward networks cannot? One possibility is that beyond a certain depth, the rank of the solution space reduces so much that good solutions no longer exist. In other words, the implicit rank-regularization of depth may take priority over the fit to training data. Residual connections are essentially "skip connections" that can be expressed as $W \to W + \mathbf{I}$, where $\mathbf{I}$ is the identity matrix (Dirac tensor for convolutions). There are two interpretations of what these connections do: one is that identity preservation prevents the rank-collapse of the solution space. The other interpretation is that residual connections reduce the *effective depth* — the number of linear operators from the input to the output (e.g., ResNet50 and ResNet101 have the same effective depth), which prevents rank-collapse of the solution space. Results in Figure 7.10 confirm this intuition. ResNets, unlike linear networks, *do not* exhibit a monotonic rank contracting behavior and the effective rank plateaus after 8 layers, regardless of using batch-normalization or not. Furthermore, preliminary experiments on least-squares using linear residual networks indicate that the effective rank of the solution space is also bounded by the number of layers in the shortest and longest path from the inputs to the outputs. A thorough study on the relationship between residual connections and rank is left for future work.

7.9 Least-squares learning dynamics

The learning dynamics of a linear network change when over-parameterized. Here, we derive the effective update rule on least-squares using linear neural networks to provide motivation on why they have differing update dynamics. For a single-layer linear network parameterized by W, without bias, the update rule is:

$$W^{(t+1)} \leftarrow W^{(t)} - \eta \nabla_{W^{(t)}} \mathcal{L}(W^{(t)}, x, y) \tag{7.11}$$

$$= W^{(t)} - \eta \nabla_{W^{(t)}} \frac{1}{2}(y - W^{(t)}x)^2 \tag{7.12}$$

$$= W^{(t)} - \eta(W^{(t)}xx^T - yx^T) \tag{7.13}$$

Where η is the learning rate. Similarly, the update rule for the two-layer network $y = W_e x = W_2 W_1 x$ can be written as:

$$W_1^{(t+1)} \leftarrow W_1^{(t)} - \eta(W_2^{(t)})^T(W_e^{(t)}xx^T - yx^T) \tag{7.14}$$

$$W_2^{(t+1)} \leftarrow W_2^{(t)} - \eta(W_e^{(t)}xx^T - yx^T)(W_1^{(t)})^T \tag{7.15}$$

$$\tag{7.16}$$

Using a short hand notation for $\nabla\mathcal{L}^{(t)} = W_e^{(t)}xx^T - yx^T$, we can compute the effective update rule for the two-layer network:

$$W_e^{(t+1)} = W_2^{(t+1)}W_1^{(t+1)} \tag{7.17}$$

$$= W_e^{(t)} - \overbrace{\eta(W_2^{(t)}W_2^{(t)T}\nabla\mathcal{L}^{(t)} + \nabla\mathcal{L}^{(t)}W_1^{(t)T}W_1^{(t)})}^{\text{first order } \mathcal{O}(\eta)} + \overbrace{\eta^2\nabla\mathcal{L}^{(t)}W_e^{(t)T}\nabla\mathcal{L}^{(t)}}^{\text{second order } \mathcal{O}(\eta^2)} \tag{7.18}$$

$$\approx W_e^{(t)} - \eta(P_2\nabla\mathcal{L}^{(t)} + \nabla\mathcal{L}^{(t)}P_1^T) \tag{7.19}$$

Where $P_i^{(t)} = W_i^{(t)}W_i^{(t)T}$ are the preconditioning matrices. The higher order terms can be ignored if the step-size is chosen sufficiently small.

(General case) For a linear network with d-layer expansion, the update for layer $1 \leq i \leq d$ is:

$$W_i^{(t+1)} \leftarrow W_i^{(t)} - \eta \overbrace{(W_d^{(t)} \cdots W_{i+1}^{(t)})^T}^{\text{weights} > i} \overbrace{(W_e^{(t)}xx^T - yx^T)}^{\text{original gradient}} \overbrace{(W_{i-1}^{(t)} \cdots W_1^{(t)})^T}^{\text{weights} < i} \tag{7.20}$$

Denoting $W_{j:i} = W_j \cdots W_{i+1} W_i$ for $j > i$, the effective update rule for the end-to-end matrix is:

$$W_e^{(t+1)} = \prod_{1<i<d} W_i^{(t+1)} = \prod_{1<i<d} (W_i - \eta W_{d:i+1}^{(t)T} \nabla \mathcal{L}^{(t)} W_{i-1:1}^T) \tag{7.21}$$

$$= W_e^{(t)} - \eta \sum_{1<i<d} W_{d:i+1} W_{d:i+1}^T \nabla \mathcal{L}^{(t)} W_{i-1:1}^T W_{i-1:1} + \mathcal{O}(\eta^2) + \cdots + \mathcal{O}(\eta^d) \tag{7.22}$$

$$\approx W_e^{(t)} - \eta \sum_{1<i<d} \underbrace{W_{d:i+1} W_{d:i+1}^T}_{\text{left precondition}} \underbrace{\nabla \mathcal{L}^{(t)}}_{\text{original gradient}} \underbrace{W_{i-1:1}^T W_{i-1:1}}_{\text{right precondition}} \tag{7.23}$$

The update rule for the general case has a much more complicated interaction of variables. For the edge $i = 1$ and $i = p$ the left and right preconditioning matrix is an identity matrix respectively.

7.10 Rank-landscape

We visualize the effective rank landscape of the effective weights in Figure 7.11 and Gram matrices in Figure 7.12. We use single and two-layer linear networks for effective-rank landscape. We use two-layer, and four-layer ReLU networks for the Gram matrix and are constructed from 128 randomly sampled input data. For both methods, all the weights are sampled from the same distribution. The landscape is constructed by moving along random directions u, v. We observe that over-parameterized linear and non-linear models almost always exhibit a lower-rank landscape than their shallower counterparts.

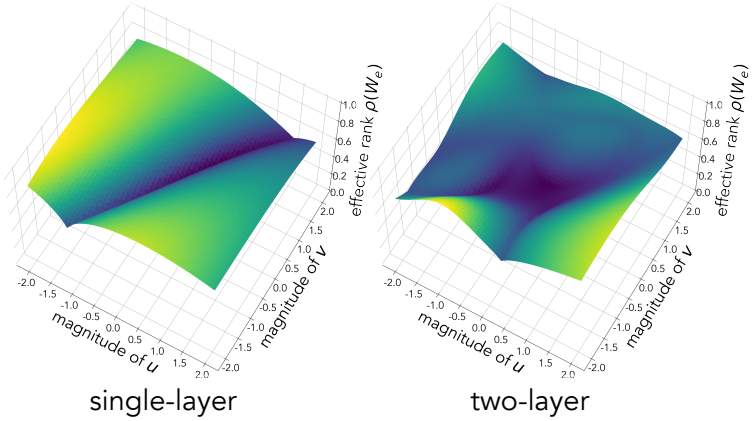


Figure 7.11: **Rank landscape**: The landscape of the effective rank ρ of a linear function W_e parameterized either by a single-layer network ($W_e = W$) or a two-layer linear network ($W_e = W_2 W_1$). The visualization illustrates a simplicity bias of depth, where the two-layer model has relatively more parameter volume mapping to lower rank W_e. Both models are initialized to the same end-to-end weights W_e at the origin. Motivated by (Goodfellow, Vinyals, and Saxe 2014), the landscapes are generated using 2 random parameter directions u, v to compute $f(\alpha, \beta) = \rho(W + \alpha \cdot u + \beta \cdot v)$ for the single-layer model and $f(\alpha, \beta) = \rho((W_2 + \alpha \cdot u_2 + \beta \cdot v_2) \cdot (W_1 + \alpha \cdot u_1 + \beta \cdot v_1))$ for the two-layer model ($u = [u_1, u_2]$, $v = [v_1, v_2]$).

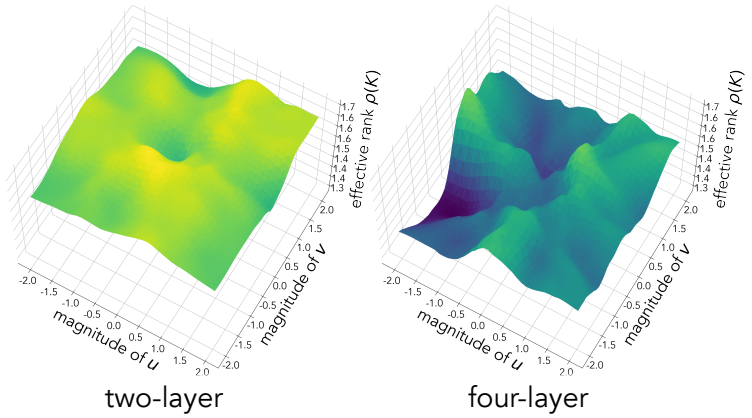


Figure 7.12: **Kernel rank landscape**: The landscape of the effective rank ρ computed on the kernels constructed from random features.

7.11 Relationship between weight and embeddings

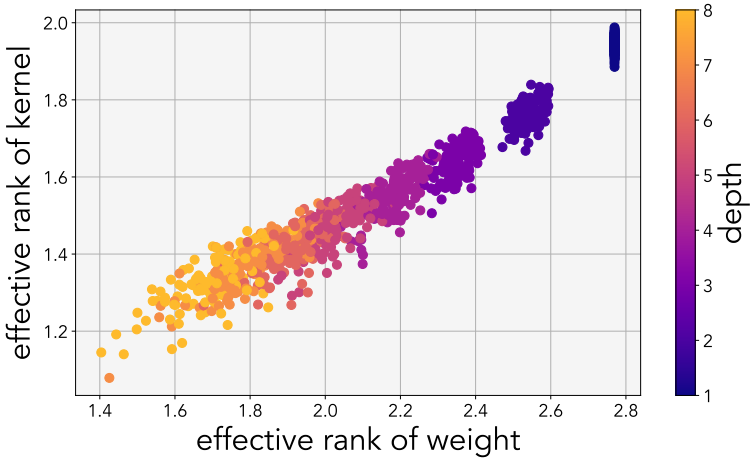


Figure 7.13: **Rank relation of kernel and weight**: Each point represents randomly drawn network. For each network, we compute the rank on the effective weight and also the linear kernel. The kernel is constructed from the `MNIST` dataset. The rank of the kernels and weights have a linear relationship.

We show that there is an almost one-to-one relationship between the effective rank of the weights and the effective rank of the Gram matrices in deep linear models. The figure plots this relationship for random deep linear networks applied to random subsets of the `MNIST` dataset. Moreover, it becomes apparent that the number of layers dictates the rank of the embedding as well as the weights.

7.12 Noisy linear regression with least-squares

We extend our least-squares analysis under noisy observation to study the interaction between noise and the low-rank bias of the deep networks. We consider the standard linear regression with the least-squares objective with additive Gaussian noise:

$$Y = WX + \mathcal{N}(\mathbf{0}, \sigma \cdot \mathbf{I}) \tag{7.24}$$

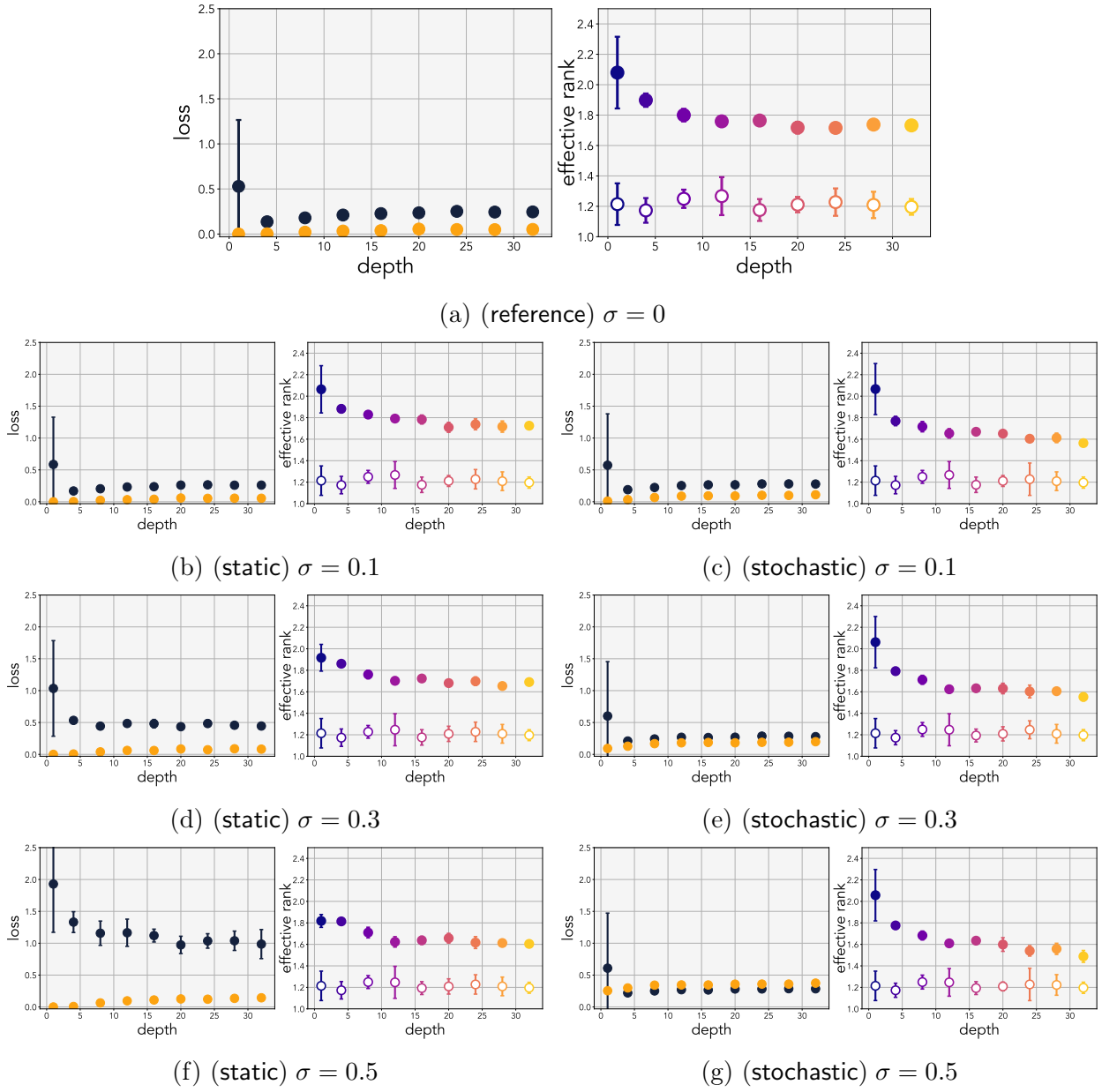
In noisy linear regression, even if the intrinsic dimensionality of W is low, the noise in the observation makes the relation between X and Y appear as full-rank. We consider two instantiations of noise injection, one in which the noise is sampled once (`static`) and another in which noise is resampled every iteration (`stochastic`). While the training loss is noisy, the test loss is computed on samples drawn from a noiseless system. In the presence of low-rank bias, even under noisy observations, deeper networks should be biased toward finding a low-rank solution. Hence, deeper networks should not overfit to the noise and result in better generalization.

To test this hypothesis, we repeat the least-squares experiment under noisy observation (see Figure 7.14). We set $\text{rank}(W) = 24$, for $W \in \mathbb{R}^{32\times 32}$ and add varying levels of noise $\sigma \in \{0.1, 0.3, 0.5\}$ to the training labels. For all varying depths, we initialize the network to the same distribution and train the network with SGD and a learning-rate scheduler (decay-on-plateau). Note that the y-axis is scaled higher than the figures in the main paper to ensure we use the same y-axis across different noise levels. For all noise levels, we observed that deeper networks converge towards low-rank solutions, and we find that the sweet-spot depth that yields the best test performance varies for each setting. The experiments yielded a few surprising observations:

1. For static additive noise, the noise *can* be overfitted by the model. In this setting, we observed that shallower networks perfectly overfit to the noise while deeper networks cannot. Unlike the noiseless least squares, deeper networks resulted in better test performance. The shallower networks find solutions that are much higher effective rank. The observation implies that depth regularizes the model from overfitting to the noise.

2. For stochastic additive noise, the noise *cannot* be overfitted by the model. In this setting, we observed that the deeper networks found an even lower effective rank solution than the noiseless counterpart. Ultimately, while shallower networks perform worse than their noise-less counterpart, the deeper networks perform on par or better. We hypothesize that the stochasticity and simplicity bias leads to lower effective rank solutions.

In both settings of noisy least-squares, we observed that the simplicity bias of depth still persists. We observed that depth improves generalization performance by underfitting the noise in the data. This may explain why deep networks generalize well under weak supervision and corrupted labels. These observations further suggest that under noisy data, one should increase the depth to mitigate overfitting to noise.



(a) (reference) $\sigma = 0$

(b) (static) $\sigma = 0.1$

(c) (stochastic) $\sigma = 0.1$

(d) (static) $\sigma = 0.3$

(e) (stochastic) $\sigma = 0.3$

(f) (static) $\sigma = 0.5$

(g) (stochastic) $\sigma = 0.5$

Figure 7.14: **Noisy least-squares:** Experiments investigating how noise affects the simplicity bias of depth.

Chapter 8

Chapter 4 Appendix

8.1 Training details

Training details: We adhere to the standard training protocols. Below are the references:

Vision training code	PyTorch TorchVision references
ViT implementation	PyTorch TorchVision models
MLP-mixer implementations	huggingface/pytorch-image-models
Quantization	TimDettmers/bitsandbytes

We replace the fused linear layers with standard linear layers to use LoRA. LoRA is applied across all linear layers. All experiments incorporate mixed-precision training. For nodes equipped with 4 GPU devices, we implement gradient checkpointing. We used gradient checkpointing for ViT-B and ViT-L.

Hardware: Our experiments were conducted using various NVIDIA GPUs, including V100 and Titan RTX.

Architecture details:

Architecture	ViT-S	ViT-B	ViT-L
Patch-sizse	32	32	32
Attention blocks	12	12	24
Attention heads	6	12	16
Hidden dim	6	768	1024
MLP dim	1536	3072	4096
Total parameters	22.9M	88.2M	306.5M

Table 8.1: ViT architecture details.

Architecture	Mixer-T	Mixer-S	Mixer-B
Patch-sizs	32	32	32
Mixer blocks	6	8	12
Embed dim	384	512	768
MLP dim	1536	2048	3072
Total parameters	8.8M	19.1M	60.3M

Table 8.2: MLP-mixer architecture details

Architecture	NanoGPT	GPT2
Block-size	256	1024
Attention blocks	6	12
Attention heads	6	12
Hidden dim	384	768
MLP dim	1152	2304
Total parameters	10.7M	124.4M

Table 8.3: LLM GPT architecture details.

Dataset details:

Dataset	CIFAR10	CIFAR100	STL10	CALTECH256	SUN397	ImageNet100	ImageNet1K
orig size	32×32	32×32	96×96	Variable	Variable	Variable	Variable
train size	224×224	224×224	224×224	224×224	224×224	224×224	224×224
num cls	10	100	10	257	397	100	$1,000$
num ims	$60,000$	$60,000$	$13,000$	$30,607$	$108,754$	$130K$	>1.2M
lr η_{default}	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-4}$	$3 \cdot 10^{-3}$	$3 \cdot 10^{-3}$	$3 \cdot 10^{-3}$
lr η_{lte}	$2 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$2 \cdot 10^{-5}$	$5 \cdot 10^{-6}$	$3 \cdot 10^{-5}$	$3 \cdot 10^{-5}$
Batch-size	1024	1024	1024	1024	1024	4096	8192

Table 8.4: Specifications for vision datasets. Most dataset images indicated as "variable size" are larger than the training image size. We provide training configuration used for ViT. For MLP-Mixer on ImageNet100, we use a learning rate of 0.001 for full-model pre-training and $1 \cdot 10^{-4}$ for LTE, both with a batch size of 4096.

Dataset	Shakespeare	TinyStories
Total number of tokens	$1.0M$	$474.0M$
Tokenizer size	65	50304
Learning-rate η_{default}	$1 \cdot 10^{-3}$	$6 \cdot 10^{-4}$
Learning-rate η_{lte}	$2 \cdot 10^{-6}$	$5 \cdot 10^{-5}$
Batch-size	512	512
Block-size	256	1024

Table 8.5: Specifications for LLM datasets and hyper-parameters used for miniGPT on Shakespeare and GPT2 on Tinystories.

LTE Optimization Details: We use $\alpha = 4096 \sim 8192$, which is $s = 128 \sim 256$ when $r = 32$ and $s = 64 \sim 128$ when $r = 64$. A good rule of thumb for the learning rate is to set it

at approximately $0.1 \sim 0.05\times$ the standard model training learning rate. We use the same learning rate scheduler as the standard pre-training, which is cosine learning-rate decay with linear warmup.

LTE Batching Detail: We use a fixed cumulative batch size for LTE. This means that given a batch size B with N LoRA heads, each head receives a batch size of $[B/N]$. When counting the training iterations, we count B and not $[B/N]$. Counting using $[B/N]$ would significantly inflate our numbers, but in the federated learning community, it is referred to as "synchronization steps". LTE training epochs were set to $4\times$ the cumulative batch size, and we exited early when we matched the performance of full-model training. For smaller datasets, our method seemed to consistently outperform the baseline, likely due to the regularization properties of rank and over-parameterization.

LTE Implementation Details: We implemented LTE using Parameter Server, model-parallelism, and DDP. Parameter server is theoretically more beneficial for our method, we conducted most of our development using DMP and DDP as it does not require rewriting communication logic. We utilize '`torch.vmap`' and simulate multiple devices on the same GPU, and share computation of the main parameters to reduce run-time.

LTE for Convolution, Affine, and Embedding Layers: Specific choices for all these layers did not impact the final performance significantly, but we detail the choices we made below.

For convolution layers, we use the over-parameterization trick in (Huh, Mobahi, Zhang, et al. 2023), which uses 1×1 for the second layer. Since convolution layers are typically used at most once in the models we tested, we did not explore beyond this parameterization. However, other potential choices exist for low-rank parameterization of convolution layers, such as channel-wise convolution and separable convolutions.

There is no notion of low-rank decomposition for affine parameters used in normalization layers. We tried various strategies to train and communicate these parameters. For affine parameters, we tried: (1) LoRA-style vector-vector parameterization $\mathbf{a}, \mathbf{b}$, (2) LoRA-style vector-scalar parameterization $\mathbf{A} = \mathbf{a}, b$, (3) DDP-style averaging, and (4) removing affine parameters. Removing affine parameters improved both baseline and LTE for classification.

Lastly, we found that using the standard averaging technique or allowing only one model to train the embedding layer worked best for the embedding layer. We chose to use standard averaging at the same iteration as the rest of the LoRA layers.

8.2 Getting exact equivalence

The exact equivalence condition for LTE and MHLoRA is achieved when the LoRA parameters are not reset. Empirically, this does lead to slightly better model performance but requires a more involved calculation. We posit that the slight improvement comes from the fact that we do not have to re-learn the LoRA parameters and ensure the optimizer state is consistent with respect to its parameters. We provide the exact equivalence below.

Denote t as the synchronization steps, τ as the local optimization step:

$$\underset{\mathbf{B}_0,\mathbf{A}_0,\ldots\mathbf{B}_N,\mathbf{A}_N}{\text{optimize}} \quad \mathbf{W}\mathbf{x} + \frac{s}{N}\sum_{i=1}^{N}\mathbf{B}_i^{(t,\tau)}\mathbf{A}_i^{(t,\tau)}\mathbf{x}$$

To match the gradient dynamics exactly, one needs to merge all LoRA and subtract the contribution of its weight after the merge. Denote $\mathbf{V}$ as the previous LoRA parameter contribution and is set to $\mathbf{0}$. Then, the LTE optimization with the correction term is:

(For each $\mathbf{B}_i, \mathbf{A}_i$ optimize for τ steps)

$$\mathbf{W}^{(t)}\mathbf{x} - s\mathbf{V}_i^{(t)}\mathbf{x} + s\mathbf{B}_i^{(t,0)}\mathbf{A}_i^{(t,0)}\mathbf{x}$$

(Merge all $\mathbf{B}_i, \mathbf{A}_i$)

$$\mathbf{W}^{(t+1)} = \mathbf{W}^{(t)} + \frac{s}{N}\sum_{i=1}^{N}\left(\mathbf{B}_i^{(t,\tau)}\mathbf{A}_i^{(t,\tau)} - \mathbf{V}_i^{(t)}\right)$$

(Update $\mathbf{V}_i$)

$$\mathbf{V}_i^{(t+1)} = \mathbf{B}_i^{(t,\tau)}\mathbf{A}_i^{(t,\tau)}$$

(Use same parameters)

$$\mathbf{B}_i^{(t+1,0)}, \mathbf{A}_i^{(t+1,0)} = \mathbf{B}_i^{(t,\tau)}, \mathbf{A}_i^{(t,\tau)}$$

Where the equivalence to multi-head LoRA holds when $\tau = 1$. Note that the subtracting of the previous contribution can be absorbed into the weight $(\mathbf{W}^{(t)} - s\mathbf{V}_i^{(t)})\mathbf{x}$.

8.3 Merge with least-squares

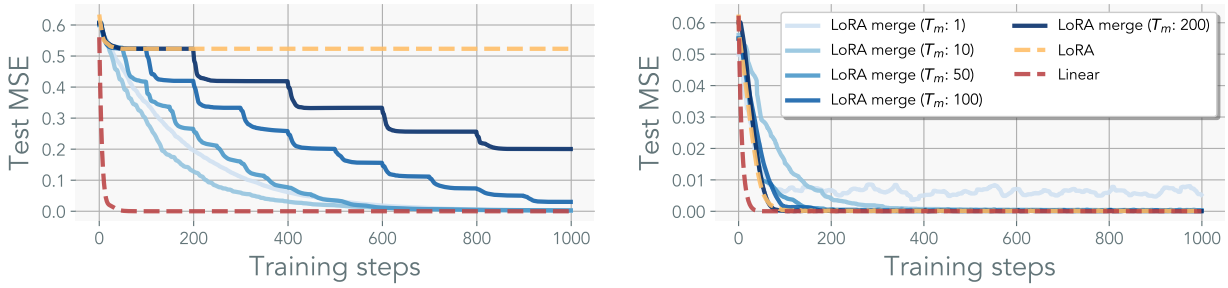


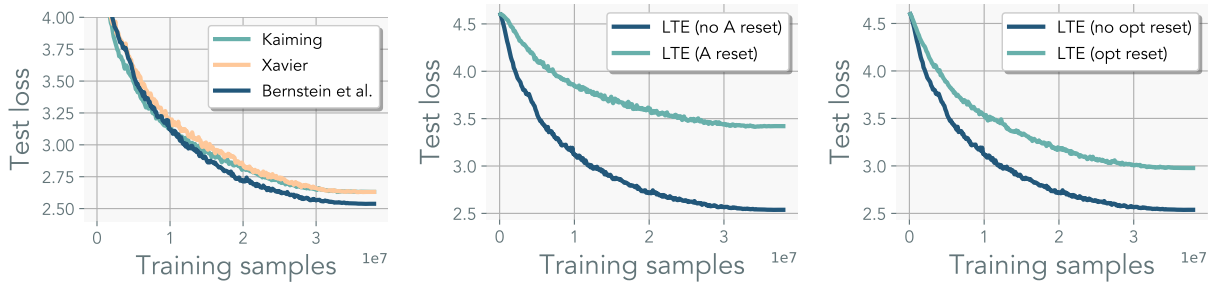
Figure 8.1: **Least-squares with LoRA**: Linear models parameterized with LoRA with varying target rank. Least-squares with $\mathbb{R}^{32\times32}$, with target rank of 32 (**right**) and 8 (**left**). LoRA is parameterized with rank $r = 4$. With merging, the model can recover the solution, with convergence scaling with merge frequencies.

We train a linear network, parameterized with LoRA, on least-squares regression. Here, we artificially constructed the problem to control for the underlying rank of the solution $\mathbf{W}^*$. We then constructed a dataset by randomly generating $\mathbf{Y} = \mathbf{X}(\mathbf{W}^*)^\mathsf{T}$. Where for each element $\mathbf{x} \in \mathbf{X}$ is drawn from a normal distribution $\mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.

Figure 8.1 visualizes the model's ability to recover the underlying ground-truth solution across various merge iterations T. Here, the optimal solution $\mathbf{W}^*$ is set to be full rank where $m = n = 32$. We employ a naive re-initialization strategy of initializing $\mathbf{A}$ with a uniform distribution scaled by the fan-out.

Without merging the LoRA parameters, the model's performance rapidly plateaus. In contrast, models trained with merges can eventually recover the full-rank solution, with the recovery rate of the solution scaling with the merging frequency.

8.4 Ablation



(a) Initialization (b) LoRA $\mathbf{A}$ reset (c) Optimizer reset

Figure 8.2: **Ablation**: These models were trained using ViT-S with 8 heads with rank $r = 16$. (**Left**) different initialization scheme. (**Middle**) resetting $\mathbf{A}$. (**Right**) resetting optimizer states for both $\mathbf{A}$ and $\mathbf{B}$.

We conducted an ablation study focusing on initialization, resetting of LoRA $\mathbf{A}$, and resetting the optimizer for the LoRA parameters. All ablation studies presented here were conducted with an LTE with rank $r = 16$, 8 heads using ViT-S on ImageNet100.

Kaiming initialization serves as the default scheme for LoRA. The conventional Kaiming initialization is tailored for square matrices and depends solely on the input dimension. Given that LoRA parameters often manifest as wide matrices, we experimented with various initialization schemes. Xavier initialization ((Glorot and Bengio 2010)) preserves the variance relationship of a linear layer for rectangular matrices. Bernstein et al. ((Bernstein, Mingard, Huang, et al. 2023)) employ semi-orthogonal initialization to maintain the spectral norm of the input. As depicted in Figure 8.2a, the method by Bernstein et al. proved most effective. It should be noted that none of these initialization methods assume residual connection or zero-ed out LoRA parameters. Hence, further tuning of the gain parameters might be needed.

When resetting the LoRA parameters, we investigated the impact of re-initializing the matrix $\mathbf{A}$ as well as its optimizer. As shown in Figure 8.2b, we found that resetting matrix $\mathbf{A}$ adversely

affects model performance, possibly due to the necessity of re-learning the representation at each iteration and discarding the momentum states. In Figure 8.2c, we also experimented with retaining the LoRA parameters while resetting the optimizer state for those parameters. Similarly, we found that resetting the optimizer state diminishes performance.

8.5 Grassman distance of LoRA heads

Motivated by (Hu, shen, Wallis, et al. 2022), we measure the Grassman distance of the LoRA heads to measure sub-space similarity between the optimized sub-spaces. Grassman distance measures the distance of k-dimensional subspace in a n-dimensional space. The distance is defined as:

Grassman distance Given subspaces U and V, and given the singular values $U^TV = X\Sigma Y^T$, where Σ is a diagonal matrix with singular values σ_i. The principal angles θ_i between U and V are given by $\theta_i = \cos^{-1}(\sigma_i)$. Then the Grassmann distance $d_{\mathsf{Grassman}}(U, V)$ is then defined as:

$$d_{\mathsf{Grassman}}(U, V, k) = \left(\sum_{i=1}^{k} \theta_i^2\right)^{\frac{1}{2}}$$

Where k is the number of principal angles, the dimension of the smaller subspace.

For LoRA, the number of principal components is spanned by the LoRA rank r; therefore, we set $k = r$. When the LoRA parameters span the same sub-space, they have a Grassman distance of 0. The pairwise Grassman distance is measured by:

$$\frac{1}{2N} \sum_{(i,j)\in[1,N]\wedge i\neq j} d_{\mathsf{Grassman}}(f_\perp(\mathbf{B}_i\mathbf{A}_i), f_\perp(\mathbf{B}_j\mathbf{A}_j), r) \tag{8.1}$$

Where $f_\perp(\cdot)$ computes the orthogonal basis by computing the input matrix's left singular vectors.

8.6 Training curve on ImageNet1k

We plot the test curves in Figure 8.3 for both standard and LTE training on ImageNet-1K. Both models were trained using a cosine learning rate scheduler. We found that training LTE for 300 epochs performed roughly 5% worse. Hence, we repeated the experiment by setting the training epoch to 600. The final performance was matched at around 420 epochs. For LTE, we doubled the batch size to 8192 (see Appendix 8.7). We found that LTE benefits from larger cumulative batch size and does not require as intensive of data augmentation as standard training, likely due to implicit regularization induced by low-rank parameterization and stale updates. The baseline did not improve when using a bigger batch size. In simpler tasks, using bigger-batches did not perform better.

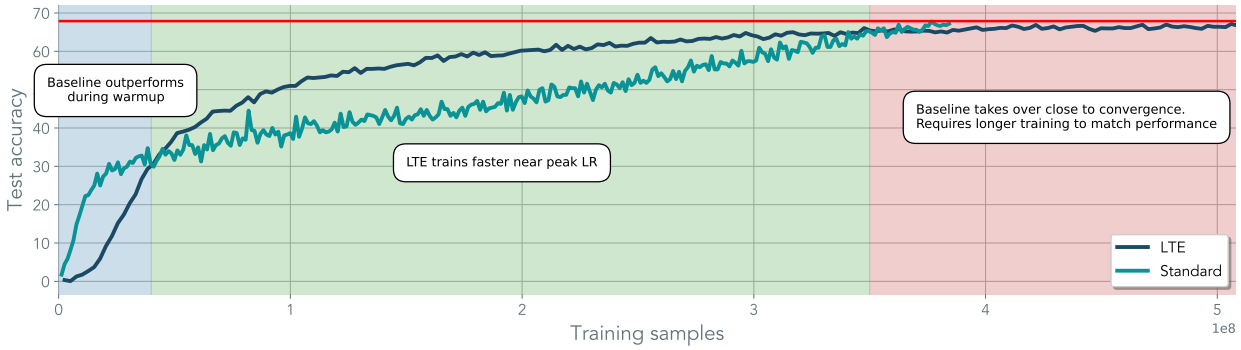


Figure 8.3: **ImageNet1k test curve**

Early in the training phase, the baseline outperformed LTE, but this trend was quickly reversed after the first initial epochs. LTE approached its final performance quite rapidly. However, LTE fell short compared to the standard training duration by 300 epochs, and an additional 120 epochs were required to reach the same test accuracy. Unlike LTE, we found that standard training benefited significantly more from the learning-rate scheduler, possibly hinting that the gradient noise and stale parameter estimates overpower the signal-to-noise ratio. We observed this trend across all ViT sizes.

A few ways to mitigate the slow convergence include synchronizing the mini-batches or the LoRA parameters as the model is trained; we leave such investigation for future works. The baseline was trained with a weight-decay of 0.01, resulting in slow convergence but better final performance – standard optimization does exhibit faster convergence if the weight-decay is removed. Training the baseline for longer, 600 epochs, did not achieve better performance.

8.7 Effect of batch-size on ImageNet1k

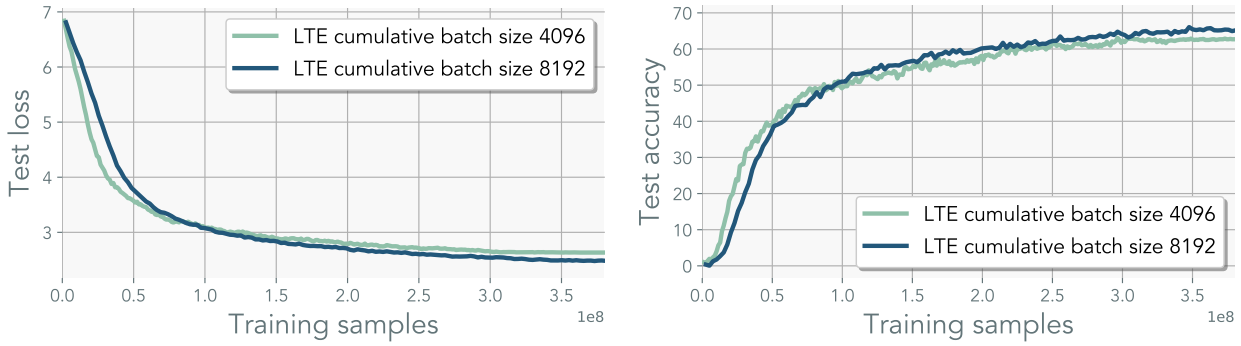


Figure 8.4: **ImageNet1k with doubled accumulative batch-size**

Utilizing larger batch sizes is beneficial for maximizing FLOP efficiency. However, when evaluated in terms of samples seen, larger batch sizes are known to underperform (Masters and Luschi 2018). Given that LTE introduces more noise, we hypothesized that a reduced learning rate could be adversely affected by both gradient noise and estimate noise from

using merges. In Figure 8.4, we experimented with increasing the batch size and observed a moderate improvement of 3% with bigger batches.

8.8 Rank of the model in pre-training

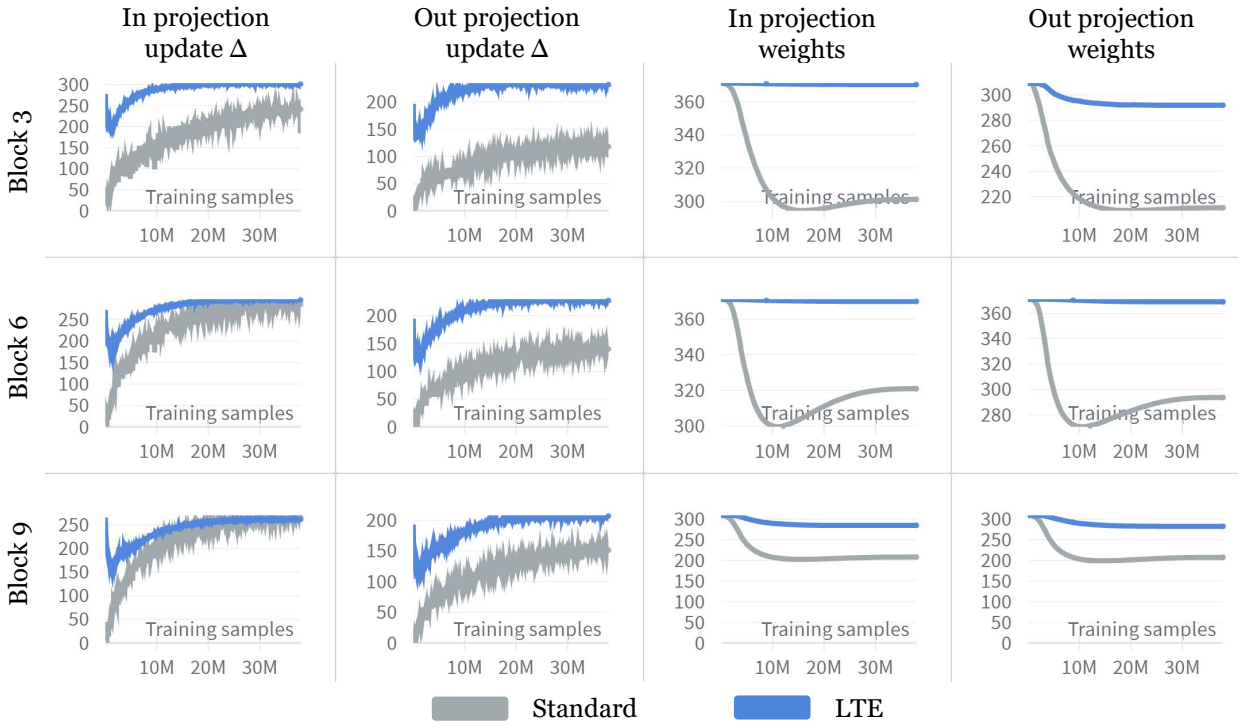


Figure 8.5: **Rank dynamics of ViT for standard training and LTE**. Rank is measured using effective rank. We track the rank of the weights and update to the main weight throughout training.

We measure the effective rank (Roy and Vetterli 2007) of standard training and LTE throughout training.

(Definition) Effective rank (spectral rank) For any matrix $A \in \mathbb{R}^{m \times n}$, the effective rank ρ is defined as the Shannon entropy of the normalized singular values:

$$\rho(A) = \exp\left(-\sum_{i=1}^{\min(n,m)} \bar{\sigma}_i \log(\bar{\sigma}_i)\right),$$

where $\bar{\sigma}_i = \sigma_i / \sum_j \sigma_j$ are normalized singular values, such that $\sum_i \bar{\sigma}_i = 1$.

The rank of the updates for standard training is the gradients, and LTE is $\nabla_{\mathbf{W}}\mathcal{L}$ and $\frac{s}{N}\sum_n \mathbf{B}_n\mathbf{A}_n$ respectively. In the context of standard training, the rank of the weights exhibits only a minor decrease throughout the optimization process. Conversely, the rank of the

gradient monotonically increases following the initial epochs. This observation provides empirical evidence that approximating the updates with a single LoRA head is not feasible. Despite its markedly different dynamics, LTE can represent full-rank updates throughout the training period. This may also be useful for designing how many LoRA heads to use with LTE, where the number of LoRA heads can start with one and slowly annealed to match the maximum rank of the weights.

8.9 Is scaling s the same as scaling the learning rate?

There is a misconception that the scalar s only acts as a way to tune the learning rate in these updates. Focusing on the update for $\mathbf{B}$ (same analysis holds for $\mathbf{A}$), we can write out the gradient as:

$$g(\mathbf{B}) = \frac{\partial \mathcal{L}}{\partial \mathbf{B}} = s\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}(\mathbf{A}\mathbf{z}_{in})^T = s\bar{g}_t \tag{8.2}$$

If we were using stochastic gradient descent, we would expect s to behave like a linear scaling on the learning rate:

$$\Delta(\mathbf{B}) = -\eta s\bar{g} \tag{8.3}$$

Where we denoted $\bar{g}$ as the component of the gradient with s factored out. We now show that s does not linearly scale the learning rate for Adam (this analysis can be extended to scale-invariant optimizers). Adam is a function of the first-order momentum m_t and second-order momentum v_t. One can factor out s from the momentum term: $m_t = s\hat{m}_t = s\beta_1\hat{m}_{t-1} + (1-\beta_1)\hat{g}_t$ and $v_t = s^2\hat{v}_t = s^2\beta_1\hat{v}_{t-1} + (1-\beta_1)\hat{g}_t^2$. Incorporating the gradient into the update rule, we see that the adaptive update does not depend linearly on s:

$$\Delta(\mathbf{B}) = -\eta\frac{m_t}{\sqrt{v_t+\epsilon}} = -\eta\frac{s\hat{m}_t}{\sqrt{s^2\hat{v}_t+\epsilon}} = -\eta\frac{\hat{m}_t}{\sqrt{\hat{v}_t+\epsilon}} \tag{8.4}$$

However, $\hat{g}_t$ is not invariant to s. Therefore, while s is not the same as the learning rate, it will impact the downward gradient $\frac{\partial \mathcal{L}(\ldots,s)}{\partial \mathbf{z}_{out}}$. We will discuss this in the next subsection. It is worth noting that s quadratically impacts the attention map and may require a separate scaling factor as opposed between MLP layers and attention layers. Consider a batched input $\mathbf{X}$ with the output of a linear as $\hat{\mathbf{X}} = \mathbf{W}\mathbf{X} + s\mathbf{B}\mathbf{A}\mathbf{X} = \mathbf{W}\mathbf{X} + s\mathbf{D}$. Then, the un-normalized attention map is dominated by the LoRA parameters:

$$\left(\mathbf{W}_Q\hat{X}\right)\left(\mathbf{W}_K\hat{X}\right)^T = \mathbf{W}_Q\left(\mathbf{W}\mathbf{X} + s\mathbf{D}\right)\left(\mathbf{X}^T\mathbf{W}^T + s\mathbf{D}^T\right)\mathbf{W}_K^T \tag{8.5}$$

$$= \cdots + s^2\mathbf{W}_Q\mathbf{D}\mathbf{D}^T\mathbf{W}_K^T \tag{8.6}$$

Unlike learning rate, scaling s affects both the forward and backward dynamics. Large s emphasizes the contribution of the LoRA parameters, which may explain why we have

observed better performance when using larger s for pre-training. It is possible that using a scheduler for s could further speed up training or even better understand how to fuse s into the optimizer or $\mathbf{A}$; we leave this for future work. Next, we dive into the effect of s on $\bar{g}_t$.

8.10 The effective update rule for LoRA is different from standard update

Effective update of LoRA. Let $\mathbf{W}$ be the original weight of the model, and denote $g(\mathbf{W}) = g$ as the gradient of the parameter. Let $\hat{\mathbf{W}} = \mathbf{W} + s\mathbf{BA}$ be the effective weight of the LoRA parameterization, and $g(\hat{\mathbf{W}}) = \hat{g}$ be its corresponding effective gradient. Then, the LoRA parameterization is related to the gradient of the standard parameterization by

$$\hat{g} = s\left(\mathbf{B}\mathbf{B}^T g - g\mathbf{A}^T\mathbf{A}\right) - s^2\eta\left(g\left(\mathbf{B}\mathbf{A}\right)^T g\right) \tag{8.7}$$

When s is small, we can safely discard the second term as it will scale quadratically with learning rate $\eta\hat{g}$. However, when s is large, the contribution of the second term becomes non-negligible. This term can be interpreted as the alignment of the LoRA parameters, and taking a step in this direction encourages $\mathbf{B}$ and $\mathbf{A}$ to be spectrally aligned. The increased contribution of the LoRA parameters and the alignment induced by larger s may explain our observation that higher s leads to better performance. It's important to note that with a learning rate scheduler, the contribution of the second term would decay to zero.

8.11 Derivation

Over-parameterization or linear re-parameterization, in general, has a non-trivial effect on the optimization dynamics. Here, we analyze the update of the effective weight to point out a rather surprising interaction between s and η. Consider a standard update rule for SGD for $\mathbf{z}_{in} \in \mathbb{R}^{n\times 1}$, and $\mathbf{z}_{out} \in \mathbb{R}^{m\times 1}$ and $\mathbf{W} \in \mathbb{R}^{m\times n}$:

$$g(\mathbf{W}) = \frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}\frac{\partial \mathbf{z}_{out}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}\mathbf{z}_{in}^T \tag{8.8}$$

We will denote $g(\mathbf{W}) = g$ from now on for clarity. For standard LoRA with parameters $\hat{\mathbf{W}} = \mathbf{W} + s\mathbf{BA}$, where $\mathbf{B} \in \mathbb{R}^{m\times r}$, $\mathbf{A} \in \mathbb{R}^{r\times n}$, the update rule on the effective weight is:

$$\hat{\mathbf{W}} \leftarrow \mathbf{W} + s\left(\mathbf{B} - \eta\frac{\partial \mathcal{L}}{\partial \mathbf{B}}\right)\left(\mathbf{A} - \eta\frac{\partial \mathcal{L}}{\partial \mathbf{A}}\right) \tag{8.9}$$

$$= \mathbf{W} + s\mathbf{BA} - s\eta\left(\left(\mathbf{B}\frac{\partial \mathcal{L}}{\partial \mathbf{A}} - \mathbf{A}\frac{\partial \mathcal{L}}{\partial \mathbf{B}}\right) + \eta\frac{\partial \mathcal{L}}{\partial \mathbf{B}}\frac{\partial \mathcal{L}}{\partial \mathbf{A}}\right) \tag{8.10}$$

We denote $g(\hat{\mathbf{W}})$ as $\hat{g}$. With the resulting effective update being:

$$\hat{g} = \left(\mathbf{B}\frac{\partial \mathcal{L}}{\partial \mathbf{A}} - \frac{\partial \mathcal{L}}{\partial \mathbf{B}}\mathbf{A}\right) - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{B}}\frac{\partial \mathcal{L}}{\partial \mathbf{A}} \tag{8.11}$$

Computing the derivative for each variable introduces the dependency on s.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{B}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}\frac{\partial \mathbf{z}_{out}}{\partial \mathbf{z}_{res}}\frac{\partial \mathbf{z}_{res}}{\partial \mathbf{B}} = s\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}(\mathbf{A}\mathbf{z}_{in})^T \tag{8.12}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{A}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}\frac{\partial \mathbf{z}_{out}}{\partial \mathbf{z}_{res}}\frac{\partial \mathbf{z}_{res}}{\partial \mathbf{A}} = s\mathbf{B}^T\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}\mathbf{z}_{in}^T \tag{8.13}$$

Plugging it back in, we have

$$\hat{g} = \left(\mathbf{B}\left(s\mathbf{B}^T\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}\mathbf{z}_{in}^T\right) - \left(s\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}(\mathbf{A}\mathbf{z}_{in})^T\right)\right)\mathbf{A} - \eta\left(s\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}(\mathbf{A}\mathbf{z}_{in})^T\right)\left(s\mathbf{B}^T\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}\mathbf{z}_{in}^T\right) \tag{8.14}$$

$$= s\left(\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}\mathbf{z}_{in}^T\mathbf{A}^T\mathbf{A} - \mathbf{B}\mathbf{B}^T\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}\mathbf{z}_{in}^T\right) - s^2\eta\left(\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}\mathbf{z}_{in}^T\mathbf{A}^T\mathbf{B}^T\frac{\partial \mathcal{L}}{\partial \mathbf{z}_{out}}\mathbf{z}_{in}^T\right) \tag{8.15}$$

$$= s\left(\mathbf{B}\mathbf{B}^T g - g\mathbf{A}^T\mathbf{A}\right) - s^2\eta\left(g\mathbf{A}^T\mathbf{B}^T g\right) \tag{8.16}$$

When s is small, both terms exist. When s is large, the second term dominates. Since the last term is quadratic with g, one can safely ignore the second term when the learning rate is sufficiently small. Similarly, when using a learning rate scheduler, the contribution of the second term would decay to zero. The second term can be interpreted as an alignment loss where the gradient moves in the direction that aligns LoRA parameters.

8.12 Method illustrations

In our illustrations, we detail the distinctions between our method and other common strategies. Distributed Data-Parallel (DDP) synchronizes the model at every iteration, with only the gradients being communicated between devices. This necessitates model synchronization across devices every iteration. Therefore, if there's a significant delay in synchronization due to slow interconnect speeds or large model sizes, synchronization becomes a bottleneck. One way to mitigate this is through local optimization, often referred to as local steps or local SGD in federated learning. Here, instead of communicating gradients, model weights are shared. Local steps are known to converge on expectation, but they still require communicating the full model, which is loaded in half or full precision, which will quickly become infeasible in 1B+ size models.

Our proposed method addresses both communication and memory issues by utilizing LoRA. Each device loads a unique set of LoRA parameters, and these parameters are updated locally. As discussed in our work, this enabled efficient exploration of full-rank updates. We

communicate only the LoRA parameters, which can be set to be the order of magnitude smaller than the original model's size. Our approach balances single-contiguous memory use with the ability to utilize more devices. The aim is to enable the training of large models using low-memory devices.

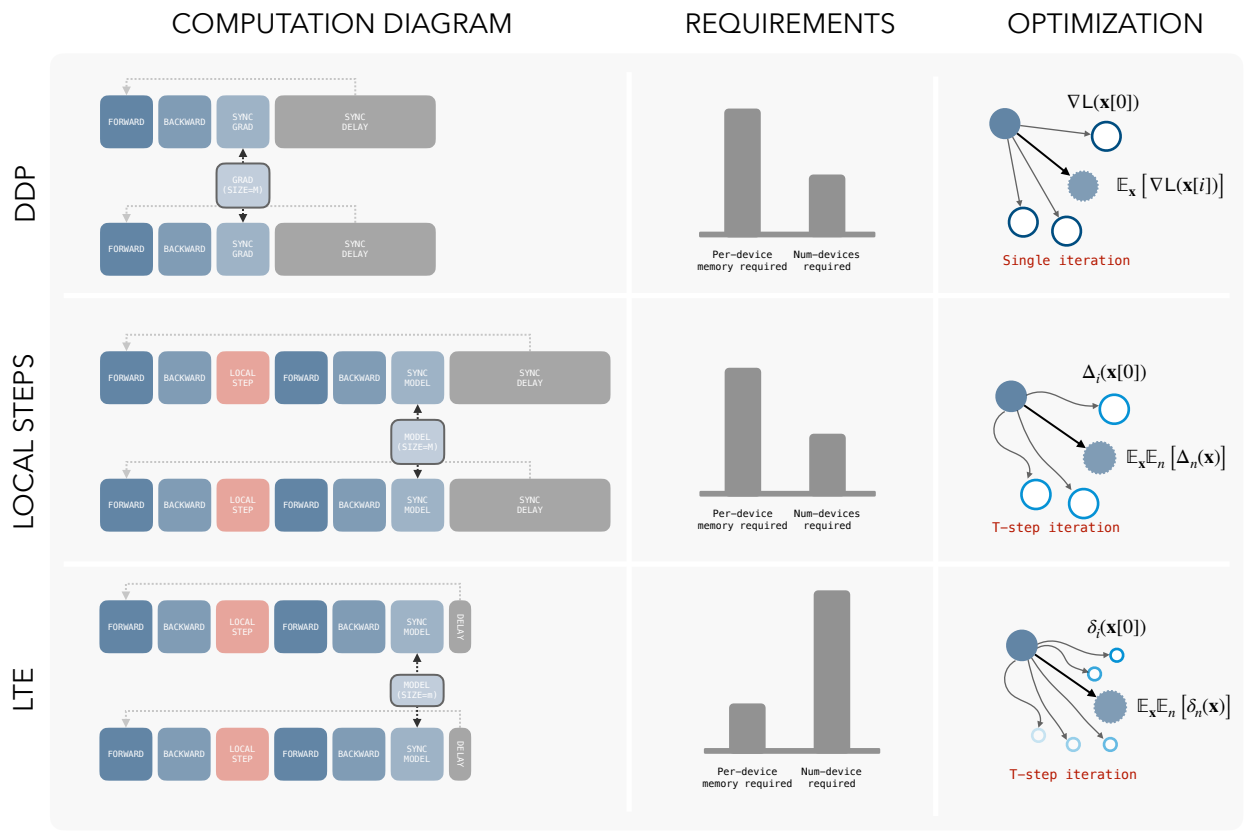


Figure 8.6: **Method illustration**: Comparisons between distributed learning methods to our method.

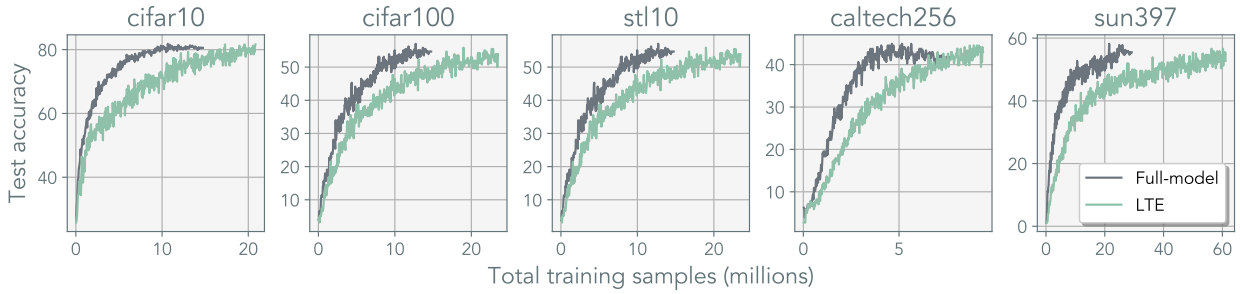
8.13 Additional results

In all of our experiments, we present two distinct curves for analysis: one that represents the total training data observed across all devices and another that shows the training samples or tokens seen per device. We use LTE with $T = 1$ for all experiments below, which is equivalent to the Multihead-LoRA optimization.

8.14 Vision Image Classification

We conduct additional experiments on CIFAR10 (Krizhevsky, Hinton, et al. 2009), CIFAR100 (Krizhevsky, Hinton, et al. 2009), STL10 (Coates, Ng, and Lee 2011), Caltech256 (Griffin, Holub, and Perona 2007), and SUN397 (Xiao, Ehinger, Hays, et al. 2016).

For these tests, we re-tuned all baseline learning rates. Detailed information about these datasets is available in Appendix 8.1. For LTE, we use ranks of $r = 64$ and $N = 32$ heads.



(a) Vision task test accuracy vs total training samples

(b) Vision task test accuracy vs training samples per device

Figure 8.7: **Additional results on various image classification datasets using ViT-S**

8.15 Scaling up ViT model size

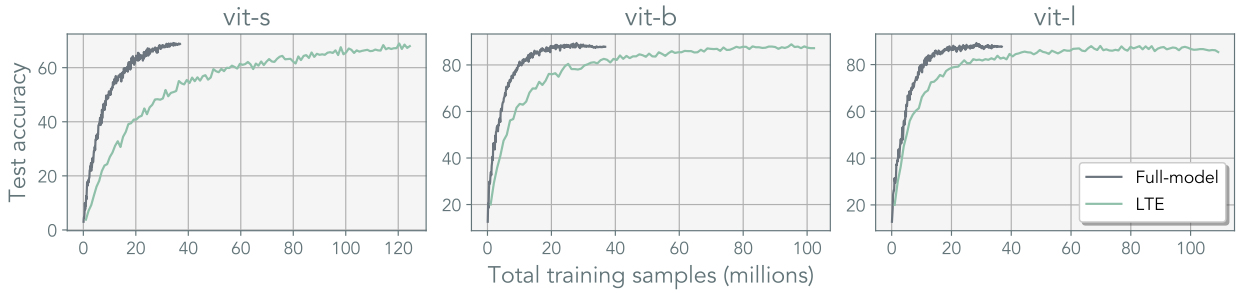
We train larger variants of the Vision Transformer (ViT) model. Details on these architectures are provided in Appendix 8.1. Across all sizes, our results remained consistent, and for ViT-L, we used a rank of $r = 128$.

8.16 LTE on MLP-Mixer

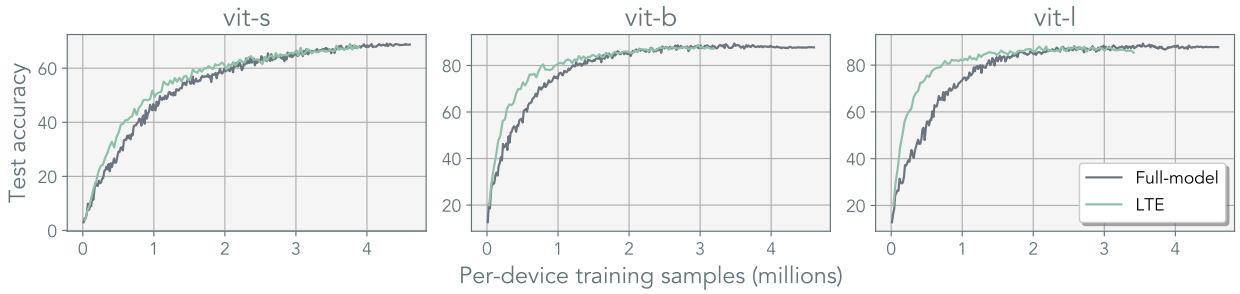
To evaluate the generalizability of our method to non-Transformer-based architectures, we train MLP-Mixer (Tolstikhin, Houlsby, Kolesnikov, et al. 2021) using LTE. The specific details of the architecture used in datasets are listed in Appendix 8.1. Our findings are consistent results across different scales of the MLP-Mixer. For Mixer-B, we use a rank of $r = 128$.

8.17 Language Modeling

We also apply our method to language modeling. For these experiments, we utilized the nanoGPT codebase (Karpathy 2023). Detailed information about the architectures and

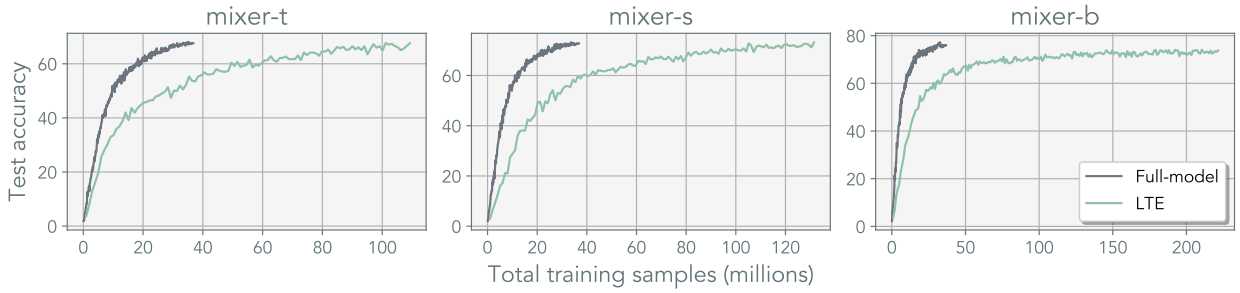


(a) ViT ImageNet100 test accuracy vs total training samples

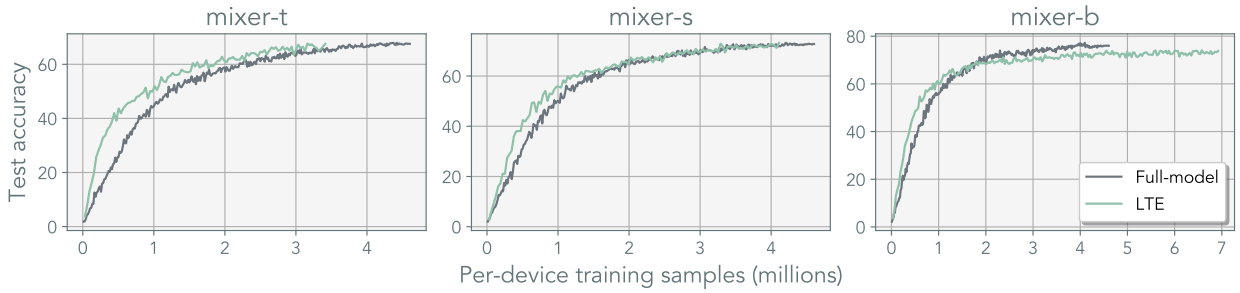


(b) ViT ImageNet100 test accuracy vs training samples per device

Figure 8.8: **ImageNet100 classification on varying ViT scale**



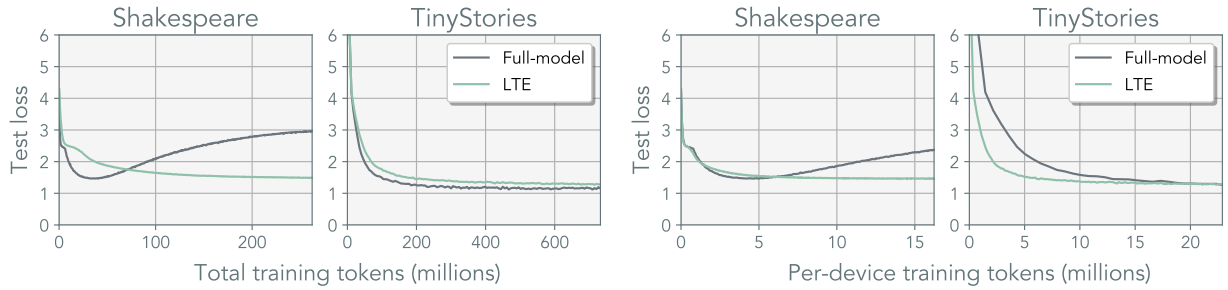
(a) MLP-mixer ImageNet100 test accuracy vs total training samples



(b) MLP-mixer ImageNet100 test accuracy vs training samples per device

Figure 8.9: **ImageNet100 classification on MLP-Mixer of varying scale**

datasets employed can be found in Appendix 8.1. Shakespeare's dataset was trained using MiniGPT (Karpathy 2023), while TinyStories (Eldan and Li 2023) was trained on GPT2 (Radford, Wu, Child, et al. 2019). For Shakespeare, we used a configuration with rank $r = 16$ and $N = 32$ heads, and for TinyStories, we employed rank $r = 64$ and $N = 32$ heads. Consistent results were observed across all sizes. We observed on simple and small datasets that LTE has a beneficial regularization effect, reducing overfitting. *Note*: We did not increase the training duration for these experiments and used fixed cumulative training samples.



(a) LLM test error vs total training samples (b) LLM test error vs training samples per device

Figure 8.10: **Additional LLM results on Shakespeare and TinyStories using GPT2**

8.18 Implementation details

We discuss a few of the implementation details we found necessary for improving the convergence speed and the performance of our method. Full training details and the supporting experiments can be found in Appendix 8.

Not resetting matrix A and optimizer states We investigate whether the matrices $\mathbf{A}_n$ would converge to the same sub-space during training. If so, it would necessitate resetting of matrices $\mathbf{A}_n$ or the use of a regularizer. In Figure 4.7, we did not observe this to be the case. We observed the orthogonality of $\mathbf{A}$ to remain consistent throughout training, and we found it to perform better without resets. We posit that re-learning matrix $\mathbf{A}$ and re-accumulating the optimizer state ends up wasting optimization steps. The comparison figures can be found in Appendix 8.4 and a more detailed discussion in Section 4.3.2.

Scaling up s and lowering learning rate η

It is a common misconception that scaling s has the same effect as tuning the learning rate η. During our experimentation, we were unable to yield comparable performance when using the standard value of s (in the range of $1 \sim 4$). Instead, we found using a large value of s and a slightly lower learning rate η to work the best. The standard practice is to set the scaling proportionately to the rank of the LoRA $s = \alpha/r$. This is done to automatically adjust for the rank (Hu, shen, Wallis, et al. 2022). We use $\alpha = 4096$ ($s = 64$) and a learning rate of $\eta = 2 \cdot 10^{-4}$. It is worth noting that the learning rate does not scale linearly with s, and the scalar only affects the forward computation (Appendix 8.9). The scalar s modifies the contribution of the LoRA parameters in the forward pass which has a non-trivial implication

on the effective gradient. Moreover, in Appendix 8.10, we provide the effective update rule of LoRA updates and observe the emergence of the term s that scales quadratically with the alignment of $\mathbf{B}$ and $\mathbf{A}$. Since s is large relative to the learning rate, it may have a non-negligible effect on the dynamics.

Initialization of LoRA plays a pivotal role in pre-training. Kaiming initialization used in the original work (Hu, shen, Wallis, et al. 2022) – are not well-suited for rectangular matrices as discussed in (Bernstein, Mingard, Huang, et al. 2023; Yang, Yu, Zhu, et al. 2024). Given that LoRA parameterization often leads to wide matrices, alternative methods from (Bernstein, Mingard, Huang, et al. 2023) and (Glorot and Bengio 2010) resulted in better empirical performance.

We use the initialization scheme prescribed in (Bernstein, Mingard, Huang, et al. 2023) that utilizes a semi-orthogonal matrix scaled by $\sqrt{d_{out}/d_{in}}$. Note that these methods were originally designed for standard feed-forward models. Whereas LoRA operates under the assumption that matrix $\mathbf{B}$ is zero-initialized with a residual connection. This aspect warrants further study for exact gain calculations. Our ablation studies, in Appendix 8.4, indicate the best performance with (Bernstein, Mingard, Huang, et al. 2023), with Kaiming and Xavier initializations performing similar. In ImageNet-1k, we found the performance gap to be more evident.

References

[1] William of Ockham. *Summa Totius Logicae*. Original work published circa 1323. Gualteri Burlei Edition, 1639.

[2] Murray Gell-Mann. *The Quark and the Jaguar: Adventures in the Simple and the Complex*. New York: W. H. Freeman and Company, 1994.

[3] Andrey N. Kolmogorov. "Three Approaches to the Quantitative Definition of Information". In: *Problems of Information Transmission* 1 (1965), pp. 1–7.

[4] Ray J. Solomonoff. "A Formal Theory of Inductive Inference. Part I and II". In: *Information and Control* 7 (1964), pp. 1–22, 224–254.

[5] Jorma Rissanen. "Modeling by Shortest Data Description". In: *Automatica* 14 (1978), pp. 465–471.

[6] Jürgen Schmidhuber. "Simple algorithmic principles of discovery, subjective beauty, selective attention, curiosity & creativity". In: *International conference on discovery science*. Springer. 2007, pp. 26–38.

[7] Marcus Hutter. *The Hutter Prize for Lossless Compression of Human Knowledge*. http://prize.hutter1.net/. http://prize.hutter1.net/. 2006.

[8] Rosa Cao and Daniel Yamins. "Explanatory models in neuroscience: Part 2–constraint-based intelligibility". In: *arXiv preprint arXiv:2104.01489* (2021).

[9] Yamini Bansal, Preetum Nakkiran, and Boaz Barak. "Revisiting model stitching to compare neural representations". In: *Advances in neural information processing systems* 34 (2021), pp. 225–236.

[10] Ronen Eldan and Ohad Shamir. "The power of depth for feedforward neural networks". In: *Conference on learning theory*. PMLR. 2016, pp. 907–940.

[11] Stuart Geman, Elie Bienenstock, and René Doursat. "Neural networks and the bias/variance dilemma". In: *Neural computation* 4.1 (1992), pp. 1–58.

[12] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. "Understanding deep learning (still) requires rethinking generalization". In: *Communications of the ACM* 64.3 (2021), pp. 107–115.

[13] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". In: *Proceedings of the National Academy of Sciences* 116.32 (2019), pp. 15849–15854.

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *arXiv preprint arXiv:2010.11929* (2020).

[15] Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. "Reconciling modern machine learning practice and the bias-variance trade-off". In: *arXiv preprint arXiv:1812.11118* (2018).

[16] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, et al. "Deep double descent: Where bigger models and more data hurt". In: *arXiv preprint arXiv:1912.02292* (2019).

[17] Suriya Gunasekar, Blake E Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nati Srebro. "Implicit regularization in matrix factorization". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6151–6159.

[18] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. "Implicit regularization in deep matrix factorization". In: *Advances in Neural Information Processing Systems* 32 (2019).

[19] Guillermo Valle-Perez, Chico Q Camargo, and Ard A Louis. "Deep learning generalizes because the parameter-function map is biased towards simple functions". In: *International Conference on Learning Representations*. 2019.

[20] Andrew M Saxe, James L Mcclelland, and Surya Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural network". In: *In International Conference on Learning Representations*. Citeseer. 2014.

[21] Madhu S Advani, Andrew M Saxe, and Haim Sompolinsky. "High-dimensional dynamics of generalization error in neural networks". In: *Neural Networks* 132 (2020), pp. 428–446.

[22] Sanjeev Arora, Nadav Cohen, and Elad Hazan. "On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization". In: *ICML*. 2018.

[23] Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. "Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning". In: *arXiv preprint arXiv:2012.09839* (2020).

[24] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. "Benign overfitting in linear regression". In: *Proceedings of the National Academy of Sciences* 117.48 (2020), pp. 30063–30070.

[25] Peter L Bartlett, Andrea Montanari, and Alexander Rakhlin. "Deep learning: a statistical viewpoint". In: *arXiv preprint arXiv:2103.09177* (2021).

[26] Sefi Bell-Kligler, Assaf Shocher, and Michal Irani. "Blind super-resolution kernel estimation using an internal-gan". In: *Advances in Neural Information Processing Systems*. 2019.

[27] Stephen Tu, Ross Boczar, Max Simchowitz, Mahdi Soltanolkotabi, and Ben Recht. "Low-rank solutions of linear matrix equations via procrustes flow". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 964–973.

[28] Cong Ma, Kaizheng Wang, Yuejie Chi, and Yuxin Chen. "Implicit regularization in nonconvex statistical estimation: Gradient descent converges linearly for phase retrieval and matrix completion". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 3345–3354.

[29] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. "Algorithmic regularization in over-parameterized matrix sensing and neural networks with quadratic activations". In: *Conference On Learning Theory*. PMLR. 2018, pp. 2–47.

[30] Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. "Implicit regularization of discrete gradient dynamics in linear neural networks". In: *Advances in Neural Information Processing Systems*. 2019, pp. 3202–3211.

[31] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. "The emergence of spectral universality in deep networks". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2018, pp. 1924–1932.

[32] Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. "In search of the real inductive bias: On the role of implicit regularization in deep learning". In: *International conference on machine learning*. 2015.

[33] Eshaan Nichani, Adityanarayanan Radhakrishnan, and Caroline Uhler. "Do Deeper Convolutional Networks Perform Better?" In: *arXiv preprint arXiv:2010.09610* (2020).

[34] Anna Golubeva, Behnam Neyshabur, and Guy Gur-Ari. "Are wider nets better given the same number of parameters?" In: *International Conference on Learning Representations*. 2021.

[35] Joel Hestness, Sharan Narang, Newsha Ardalani, et al. "Deep learning scaling is predictable, empirically". In: *arXiv preprint arXiv:1712.00409* (2017).

[36] Jared Kaplan, Sam McCandlish, Tom Henighan, et al. "Scaling laws for neural language models". In: *arXiv preprint arXiv:2001.08361* (2020).

[37] Li Jing, Jure Zbontar, et al. "Implicit Rank-Minimizing Autoencoder". In: *Advances in Neural Information Processing Systems*. Vol. 33. 2020.

[38] Shuxuan Guo, Jose M Alvarez, and Mathieu Salzmann. "ExpandNets: Linear Over-parameterization to Train Compact Convolutional Networks". In: *Advances in Neural Information Processing Systems*. 2020.

[39] Mohammad Pezeshki, Sékou-Oumar Kaba, Yoshua Bengio, et al. "Gradient Starvation: A Learning Proclivity in Neural Networks". In: *arXiv preprint arXiv:2011.09468* (2020).

[40] Aristide Baratin, Thomas George, César Laurent, et al. "Implicit regularization via neural feature alignment". In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 2269–2277.

[41] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. "The implicit bias of gradient descent on separable data". In: *The Journal of Machine Learning Research* 19.1 (2018), pp. 2822–2878.

[42] Mor Shpigel Nacson, Jason Lee, Suriya Gunasekar, et al. "Convergence of gradient descent on separable data". In: *The 22nd International Conference on Artificial Intelligence and Statistics*. PMLR. 2019, pp. 3420–3428.

[43] Suriya Gunasekar, Jason D Lee, Daniel Soudry, and Nati Srebro. "Implicit bias of gradient descent on linear convolutional networks". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9461–9471.

[44] Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, et al. "SGD on neural networks learns functions of increasing complexity". In: *arXiv preprint arXiv:1905.11604* (2019).

[45] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, et al. "A closer look at memorization in deep networks". In: *International Conference on Machine Learning*. PMLR. 2017, pp. 233–242.

[46] Harshay Shah, Kaustav Tamuly, Aditi Raghunathan, Prateek Jain, and Praneeth Netrapalli. "The pitfalls of simplicity bias in neural networks". In: *arXiv preprint arXiv:2006.07710* (2020).

[47] Vatsal Shah, Anastasios Kyrillidis, and Sujay Sanghavi. "Minimum norm solutions do not always generalize well for over-parameterized problems". In: *stat*. Vol. 1050. 2018, p. 16.

[48] Noam Razin and Nadav Cohen. "Implicit Regularization in Deep Learning May Not Be Explainable by Norms". In: *Advances in neural information processing systems*. 2020.

[49] Olivier Roy and Martin Vetterli. "The effective rank: A measure of effective dimensionality". In: *2007 15th European signal processing conference*. IEEE. 2007, pp. 606–610.

[50] Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, et al. "Skip-thought vectors". In: *Advances in Neural Information Processing Systems*. 2015.

[51] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. "The unreasonable effectiveness of deep features as a perceptual metric". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018.

[52] Abraham Savitzky and Marcel JE Golay. "Smoothing and differentiation of data by simplified least squares procedures." In: *Analytical chemistry* 36.8 (1964), pp. 1627–1639.

[53] S.H. Friedberg, A.J. Insel, and L.E. Spence. *Linear Algebra*. Featured Titles for Linear Algebra (Advanced) Series. Pearson Education, 2003. ISBN: 9780130084514. URL: https://books.google.com/books?id=HCUlAQAAIAAJ.

[54] Lior Rokach and Oded Maimon. "Clustering methods". In: *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 321–352.

[55] Dan Hendrycks and Kevin Gimpel. "Gaussian error linear units (gelus)". In: *arXiv preprint arXiv:1606.08415* (2016).

[56] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. "Implicit neural representations with periodic activation functions". In: *Advances in Neural Information Processing Systems* 33 (2020).

[57] Yurii Nesterov. "A method for unconstrained convex minimization problem with the rate of convergence O (1/kˆ 2)". In: *Doklady an ussr*. Vol. 269. 1983, pp. 543–547.

[58] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[59] Dong C Liu and Jorge Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical programming* 45.1 (1989), pp. 503–528.

[60] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)". In: *Evolutionary computation* 11.1 (2003), pp. 1–18.

[61] Gernot Akemann, Jesper R Ipsen, and Mario Kieburg. "Products of rectangular random matrices: singular values and progressive scattering". In: *Physical Review E* 88.5 (2013), p. 052118.

[62] Gernot Akemann, Mario Kieburg, and Lu Wei. "Singular value correlation functions for products of Wishart random matrices". In: *Journal of Physics A: Mathematical and Theoretical* 46.27 (2013), p. 275205.

[63] Zdzisław Burda, Andrzej Jarosz, Giacomo Livan, Maciej A Nowak, and Artur Swiech. "Eigenvalues and singular values of products of rectangular Gaussian random matrices". In: *Physical Review E* 82.6 (2010), p. 061114.

[64] Jeffrey Pennington, Samuel S Schoenholz, and Surya Ganguli. "Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice". In: *Advances in neural information processing systems*. 2017.

[65] Thorsten Neuschel. "Plancherel–Rotach formulae for average characteristic polynomials of products of Ginibre random matrices and the Fuss–Catalan distribution". In: *Random Matrices: Theory and Applications* 3.01 (2014), p. 1450003.

[66] Christian Szegedy, Wei Liu, Yangqing Jia, et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[68] Ray J Solomonoff. "A formal theory of inductive inference. Part I". In: *Information and control* 7.1 (1964), pp. 1–22.

[69] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012), pp. 1097–1105.

[70] Stephan Hoyer, Jascha Sohl-Dickstein, and Sam Greydanus. "Neural reparameterization improves structural optimization". In: *arXiv preprint arXiv:1909.04240* (2019).

[71] Barret Zoph and Quoc V Le. "Neural architecture search with reinforcement learning". In: *International Conference on Learning Representations*. 2017.

[72] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, et al. "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time". In: *International Conference on Machine Learning*. PMLR. 2022, pp. 23965–23998.

[73] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. "Visual Instruction Tuning". In: *NeurIPS*. 2023.

[74] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, et al. "Beyond the imitation game: Quantifying and extrapolating the capabilities of language models". In: *arXiv preprint arXiv:2206.04615* (2022).

[75] OpenAI. "GPT-4 technical report". In: *arXiv preprint arXiv:2303.08774* (2023).

[76] Google. "Gemini: a family of highly capable multimodal models". In: *arXiv preprint arXiv:2312.11805* (2023).

[77] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, et al. "On the opportunities and risks of foundation models". In: *arXiv preprint arXiv:2108.07258* (2021).

[78] Danny Driess, Fei Xia, Mehdi SM Sajjadi, et al. "Palm-e: An embodied multimodal language model". In: *arXiv preprint arXiv:2303.03378* (2023).

[79] Anthony Brohan, Noah Brown, Justice Carbajal, et al. "Rt-2: Vision-language-action models transfer web knowledge to robotic control". In: *arXiv preprint arXiv:2307.15818* (2023).
[80] Jun Ma, Yuting He, Feifei Li, et al. "Segment anything in medical images". In: *Nature Communications* 15.1 (2024), p. 654.
[81] Ethan Steinberg, Ken Jung, Jason A Fries, et al. "Language models are an effective representation learning technique for electronic health record data". In: *Journal of biomedical informatics* 113 (2021), p. 103637.
[82] Plato. *The Allegory of the Cave*. Written c. 375 BC. Some Ancient Publisher, –375.
[83] W. Newton-Smith. *The Rationality of Science*. International Library of Philosophy, Psychology, and Scientific Method. Routledge & Kegan Paul, 1981. ISBN: 9780710009135.
[84] Hilary Putnam. "Three kinds of scientific realism". In: *The Philosophical Quarterly (1950-)* 32.128 (1982), pp. 195–200.
[85] Gerald Doppelt. "Reconstructing scientific realism to rebut the pessimistic meta-induction". In: *Philosophy of Science* 74.1 (2007), pp. 96–118.
[86] Clyde L Hardin and Alexander Rosenberg. "In defense of convergent realism". In: *Philosophy of Science* 49.4 (1982), pp. 604–615.
[87] Yonglong Tian, Dilip Krishnan, and Phillip Isola. "Contrastive multiview coding". In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer. 2020, pp. 776–794.
[88] Roland S Zimmermann, Yash Sharma, Steffen Schneider, Matthias Bethge, and Wieland Brendel. "Contrastive learning inverts the data generating process". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12979–12990.
[89] Jonathan Richens and Tom Everitt. "Robust agents learn causal world models". In: *ICLR* (2024).
[90] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. "Similarity of neural network representations revisited". In: *International conference on machine learning*. PMLR. 2019, pp. 3519–3529.
[91] Max Klabunde, Tobias Schumacher, Markus Strohmaier, and Florian Lemmerich. "Similarity of Neural Network Models: A Survey of Functional and Representational Measures". In: *arXiv preprint arXiv:2305.06329* (2023).
[92] Nachman Aronszajn. "Theory of reproducing kernels". In: *Transactions of the American mathematical society* 68.3 (1950), pp. 337–404.
[93] Alex J Smola and Bernhard Schölkopf. *Learning with kernels*. Vol. 4. Citeseer, 1998.
[94] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. "Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability". In: *Advances in neural information processing systems* 30 (2017).
[95] Young-Jin Park, Hao Wang, Shervin Ardeshir, and Navid Azizan. "Quantifying Representation Reliability in Self-Supervised Learning Models". In: *Conference on Uncertainty in Artificial Intelligence*. 2024.
[96] Shaul Oron, Tali Dekel, Tianfan Xue, William T Freeman, and Shai Avidan. "Best-buddies similarity—Robust template matching using mutual nearest neighbors". In: *IEEE transactions on pattern analysis and machine intelligence* 40.8 (2017), pp. 1799–1813.

[97] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. "Places: A 10 million image database for scene recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 40.6 (2017), pp. 1452–1464.

[98] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, et al. "The visual task adaptation benchmark". In: (2019).

[99] Karel Lenc and Andrea Vedaldi. "Understanding image representations by measuring their equivariance and equivalence". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 991–999.

[100] Olga Russakovsky, Jia Deng, Hao Su, et al. "Imagenet large scale visual recognition challenge". In: *International journal of computer vision* (2015).

[101] Bruno A Olshausen and David J Field. "Emergence of simple-cell receptive field properties by learning a sparse code for natural images". In: *Nature* 381.6583 (1996), pp. 607–609.

[102] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[103] Luca Moschella, Valentino Maiorca, Marco Fumero, et al. "Relative representations enable zero-shot latent space communication". In: *arXiv preprint arXiv:2209.15430* (2022).

[104] Amil Dravid, Yossi Gandelsman, Alexei A Efros, and Assaf Shocher. "Rosetta neurons: Mining the common units in a model zoo". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 1934–1943.

[105] Krishna Srinivasan, Karthik Raman, Jiecao Chen, Michael Bendersky, and Marc Najork. "Wit: Wikipedia-based image text dataset for multimodal multilingual machine learning". In: *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2021, pp. 2443–2449.

[106] Aaron Gokaslan and Vanya Cohen. *OpenWebText Corpus*. http://Skylion007.github.io/OpenWebTextCorpus. 2019.

[107] Geoffrey Roeder, Luke Metz, and Durk Kingma. "On linear identifiability of learned representations". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9030–9039.

[108] Alex Krizhevsky, Geoffrey Hinton, et al. "Learning multiple layers of features from tiny images". In: (2009).

[109] Randall Balestriero and Richard G Baraniuk. "A spline theory of deep learning". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 374–383.

[110] Leland McInnes, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction". In: *arXiv preprint arXiv:1802.03426* (2018).

[111] Leo Tolstoy. *Anna Karenina*. The Russian Messenger, 1877.

[112] Vaishnavh Nagarajan and J Zico Kolter. "Uniform convergence may be unable to explain generalization in deep learning". In: *Advances in Neural Information Processing Systems* 32 (2019).

[113] Timur Garipov, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry P Vetrov, and Andrew G Wilson. "Loss surfaces, mode connectivity, and fast ensembling of dnns". In: *Advances in neural information processing systems* 31 (2018).

[114] Ekdeep Singh Lubana, Eric J Bigelow, Robert P Dick, David Krueger, and Hidenori Tanaka. "Mechanistic mode connectivity". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 22965–23004.

[115] Samuel K Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. "Git re-basin: Merging models modulo permutation symmetries". In: *arXiv preprint arXiv:2209.04836* (2022).

[116] George Stoica, Daniel Bolya, Jakob Bjorner, Taylor Hearn, and Judy Hoffman. "ZipIt! Merging Models from Different Tasks without Training". In: *arXiv preprint arXiv:2305.03053* (2023).

[117] Keller Jordan, Hanie Sedghi, Olga Saukh, Rahim Entezari, and Behnam Neyshabur. "REPAIR: REnormalizing Permuted Activations for Interpolation Repair". In: *arXiv preprint arXiv:2211.08403* (2022).

[118] Maxime Oquab, Timothée Darcet, Theo Moutakanni, et al. *DINOv2: Learning Robust Visual Features without Supervision*. 2023.

[119] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. "Hellaswag: Can a machine really finish your sentence?" In: *arXiv preprint arXiv:1905.07830* (2019).

[120] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, et al. "Training Verifiers to Solve Math Word Problems". In: *arXiv preprint arXiv:2110.14168* (2021).

[121] Jack Merullo, Louis Castricato, Carsten Eickhoff, and Ellie Pavlick. "Linearly mapping from image to text space". In: *arXiv preprint arXiv:2209.15162* (2022).

[122] Jing Yu Koh, Ruslan Salakhutdinov, and Daniel Fried. "Grounding language models to images for multimodal inputs and outputs". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 17283–17300.

[123] Ben Sorscher, Surya Ganguli, and Haim Sompolinsky. "Neural representational geometry underlies few-shot concept learning". In: *Proceedings of the National Academy of Sciences* 119.43 (2022), e2200800119.

[124] Mayug Maniparambil, Raiymbek Akshulakov, Yasser Abdelaziz Dahou Djilali, et al. "Do Vision and Language Encoders Represent the World Similarly?" In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2024, pp. 14334–14343.

[125] Pratyusha Sharma, Tamar Rott Shaham, Manel Baradad, et al. "A Vision Check-up for Language Models". In: *arXiv preprint*. 2024.

[126] James Betker, Gabriel Goh, Li Jing, et al. "Improving image generation with better captions". In: *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf* 2.3 (2023), p. 8.

[127] Long Lian, Boyi Li, Adam Yala, and Trevor Darrell. "LLM-grounded diffusion: Enhancing prompt understanding of text-to-image diffusion models with large language models". In: *arXiv preprint arXiv:2305.13655* (2023).

[128] Long Lian, Baifeng Shi, Adam Yala, Trevor Darrell, and Boyi Li. "LLM-grounded video diffusion models". In: *arXiv preprint arXiv:2309.17444* (2023).

[129] Tsung-Han Wu, Long Lian, Joseph E Gonzalez, Boyi Li, and Trevor Darrell. "Self-correcting LLM-controlled diffusion models". In: *arXiv preprint arXiv:2311.16090* (2023).

[130] Jerry Ngo and Yoon Kim. *What Do Language Models Hear? Probing for Auditory Representations in Language Models*. 2024. arXiv: 2402.16998 [cs.CL].

[131] Evonne Ng, Sanjay Subramanian, Dan Klein, et al. "Can Language Models Learn to Listen?" In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 10083–10093.

[132] Daniel LK Yamins, Ha Hong, Charles F Cadieu, et al. "Performance-optimized hierarchical models predict neural responses in higher visual cortex". In: *Proceedings of the national academy of sciences* 111.23 (2014), pp. 8619–8624.

[133] Horace B Barlow et al. "Possible principles underlying the transformation of sensory messages". In: *Sensory communication* 1.01 (1961), pp. 217–233.

[134] Bruno A Olshausen and David J Field. "Sparse coding with an overcomplete basis set: A strategy employed by V1?" In: *Vision research* 37.23 (1997), pp. 3311–3325.

[135] Richard Antonello and Alexander Huth. "Predictive coding or just feature discovery? An alternative account of why language models fit brain data". In: *Neurobiology of Language* 5.1 (2024), pp. 64–79.

[136] Colin Conwell, Jacob S Prince, Kendrick N Kay, George A Alvarez, and Talia Konkle. "What can 1.8 billion regressions tell us about the pressures shaping high-level visual representation in brains and machines?" In: *BioRxiv* (2022), pp. 2022–03.

[137] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. "A theoretical analysis of contrastive unsupervised representation learning". In: *arXiv preprint arXiv:1902.09229* (2019).

[138] Tongzhou Wang and Phillip Isola. "Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9929–9939.

[139] Yonglong Tian, Yue Wang, Dilip Krishnan, Joshua B Tenenbaum, and Phillip Isola. "Rethinking few-shot image classification: a good embedding is all you need?" In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16*. Springer. 2020, pp. 266–282.

[140] Kaiming He, Xinlei Chen, Saining Xie, et al. "Masked autoencoders are scalable vision learners. 2022 IEEE". In: *CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 15979–15988.

[141] Alec Radford, Jeffrey Wu, Rewon Child, et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[142] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[143] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. "Momentum contrast for unsupervised visual representation learning". In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 9729–9738.

[144] Alec Radford, Rafal Jozefowicz, and Ilya Sutskever. "Learning to generate reviews and discovering sentiment". In: *arXiv preprint arXiv:1704.01444* (2017).

[145] Minyoung Huh, Hossein Mobahi, Richard Zhang, et al. "The Low-Rank Simplicity Bias in Deep Networks". In: *Transactions on Machine Learning Research* (2023). ISSN: 2835-8856. URL: https://openreview.net/forum?id=bCiNWDmlY2.

[146] Kamaludin Dingle, Chico Q Camargo, and Ard A Louis. "Input–output maps are strongly biased towards simple outputs". In: *Nature communications* 9.1 (2018), p. 761.
[147] Micah Goldblum, Marc Finzi, Keefer Rowan, and Andrew Gordon Wilson. "The No Free Lunch Theorem, Kolmogorov Complexity, and the Role of Inductive Biases in Machine Learning". In: *arXiv preprint arXiv:2304.05366* (2023).
[148] Murray Gell-Mann. *The Quark and the Jaguar: Adventures in the Simple and the Complex*. Macmillan, 1995.
[149] Richard Lewis Nettleship. *Lectures on the 'Republic' of Plato*. Vol. 2. Macmillan, 1897.
[150] Paul J Werbos. "Learning how the world works: Specifications for predictive networks in robots and brains". In: *Proceedings of IEEE International Conference on Systems, Man and Cybernetics, NY*. 1987.
[151] David Ha and Jürgen Schmidhuber. "World models". In: *arXiv preprint arXiv:1803.10122* (2018).
[152] Tianyu Gao, Xingcheng Yao, and Danqi Chen. "SimCSE: Simple Contrastive Learning of Sentence Embeddings". In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2021.
[153] Yinhan Liu, Myle Ott, Naman Goyal, et al. "Roberta: A robustly optimized bert pretraining approach". In: *arXiv preprint arXiv:1907.11692* (2019).
[154] Mostafa Abdou, Artur Kulmizev, Daniel Hershcovich, et al. "Can language models encode perceptual structure without grounding? a case study in color". In: *arXiv preprint arXiv:2109.06129* (2021).
[155] Michael Gutmann and Aapo Hyvärinen. "Noise-contrastive estimation: A new estimation principle for unnormalized statistical models". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 297–304.
[156] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation learning with contrastive predictive coding". In: *arXiv preprint arXiv:1807.03748* (2018).
[157] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson. "Crisp Boundary Detection Using Pointwise Mutual Information". In: *ECCV*. 2014.
[158] Phillip Isola. "The Discovery of Perceptual Structure from Visual Co-occurrences in Space and Time". In: *MIT Ph.D. Thesis*. 2015.
[159] Phillip Isola, Daniel Zoran, Dilip Krishnan, and Edward H. Adelson. "Learning Visual Groups From Co-occurrences in Space and Time". In: *ICLR, Workshop paper*. 2016.
[160] Nathanael Chambers and Dan Jurafsky. "Unsupervised learning of narrative event chains". In: *Proceedings of ACL-08: HLT*. 2008, pp. 789–797.
[161] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).
[162] Alec Radford, Jong Wook Kim, Chris Hallacy, et al. "Learning transferable visual models from natural language supervision". In: *International conference on machine learning*. PMLR. 2021, pp. 8748–8763.
[163] Mehdi Mirza and Simon Osindero. "Conditional generative adversarial nets". In: *arXiv preprint arXiv:1411.1784* (2014).

[164] Steven Liu, Tongzhou Wang, David Bau, Jun-Yan Zhu, and Antonio Torralba. "Diverse Image Generation via Self-Conditioned GANs". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[165] Axel Sauer, Katja Schwarz, and Andreas Geiger. "StylegGAN-XL: Scaling StyleGAN to large diverse datasets". In: *ACM SIGGRAPH 2022 conference proceedings*. 2022, pp. 1–10.

[166] Tianhong Li, Dina Katabi, and Kaiming He. "Return of Unconditional Generation: A Self-supervised Representation Generation Method". In: *arXiv:2312.03701* (2023).

[167] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. "Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks". In: *Computer Vision (ICCV), 2017 IEEE International Conference on*. 2017.

[168] Yuyang Shi, Valentin De Bortoli, Andrew Campbell, and Arnaud Doucet. "Diffusion Schrödinger bridge matching". In: *Advances in Neural Information Processing Systems* 36 (2024).

[169] Shaoan Xie, Qirong Ho, and Kun Zhang. "Unsupervised image-to-image translation with density changing regularization". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 28545–28558.

[170] Dustin Tran, Yura Burda, and Ilya Sutskever. "Feature-Matching Auto-Encoders". In: (2017).

[171] Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. "Phrase-based & neural unsupervised machine translation". In: *arXiv preprint arXiv:1804.07755* (2018).

[172] John Locke. *An Essay Concerning Human Understanding*. 1690.

[173] Richard Held, Yuri Ostrovsky, Beatrice de Gelder, et al. "The newly sighted fail to match seen with felt". In: *Nature neuroscience* 14.5 (2011), pp. 551–553.

[174] Melissa Hall, Laurens van der Maaten, Laura Gustafson, Maxwell Jones, and Aaron Adcock. "A systematic study of bias amplification". In: *arXiv preprint arXiv:2201.11706* (2022).

[175] Vardan Papyan, XY Han, and David L Donoho. "Prevalence of neural collapse during the terminal phase of deep learning training". In: *Proceedings of the National Academy of Sciences* 117.40 (2020), pp. 24652–24663.

[176] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. "Pretrained transformers as universal computation engines". In: *arXiv preprint arXiv:2103.05247* 1 (2021).

[177] Suvir Mirchandani, Fei Xia, Pete Florence, et al. "Large language models as general pattern machines". In: *arXiv preprint arXiv:2307.04721* (2023).

[178] Jack Urbanek, Florian Bordes, Pietro Astolfi, et al. *A Picture is Worth More Than 77 Text Tokens: Evaluating CLIP-Style Models on Dense Captions*. 2023. arXiv: 2312.08578 [cs.CV].

[179] Meta. *Meta LLaMA 3*. 2024. URL: https://ai.meta.com/blog/meta-llama-3/.

[180] Sara Hooker. "The hardware lottery". In: *Communications of the ACM* 64.12 (2021), pp. 58–65.

[181] Ilia Sucholutsky, Lukas Muttenthaler, Adrian Weller, et al. *Getting aligned on representational alignment*. 2023. arXiv: 2310.13018 [q-bio.NC].

[182] Edward J Hu, yelong shen, Phillip Wallis, et al. "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=nZeVKeeFYf9.

[183] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. "Qlora: Efficient finetuning of quantized llms". In: *arXiv preprint arXiv:2305.14314* (2023).

[184] Arnav Chavan, Zhuang Liu, Deepak Gupta, Eric Xing, and Zhiqiang Shen. "One-for-All: Generalized LoRA for Parameter-Efficient Fine-tuning". In: *arXiv preprint arXiv:2306.07967* (2023).

[185] Qingru Zhang, Minshuo Chen, Alexander Bukharin, et al. "Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=lq62uWRJjiY.

[186] Longteng Zhang, Lin Zhang, Shaohuai Shi, Xiaowen Chu, and Bo Li. "LoRA-FA: Memory-efficient Low-rank Adaptation for Large Language Models Fine-tuning". In: *arXiv preprint arXiv:2308.03303* (2023).

[187] Vladislav Lialin, Namrata Shivagunde, Sherin Muckatira, and Anna Rumshisky. "Stack More Layers Differently: High-Rank Training Through Low-Rank Updates". In: *arXiv preprint arXiv:2307.05695* (2023).

[188] Renrui Zhang, Rongyao Fang, Wei Zhang, et al. "Tip-adapter: Training-free clip-adapter for better vision-language modeling". In: *arXiv preprint arXiv:2111.03930* (2021).

[189] Han Cai, Chuang Gan, Ligeng Zhu, and Song Han. "TinyTL: Reduce Memory, Not Parameters for Efficient On-Device Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 11285–11297. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/81f7acabd411274fcf65ce2070ed568a-Paper.pdf.

[190] Estelle Bettelli, Yijun Carrier, Wenda Gao, et al. "Reciprocal developmental pathways for the generation of pathogenic effector TH17 and regulatory T cells". In: *Nature* 441.7090 (2006), pp. 235–238.

[191] Pramod Kaushik Mudrakarta, Mark Sandler, Andrey Zhmoginov, and Andrew Howard. "K for the price of 1: Parameter-efficient multi-task and transfer learning". In: *arXiv preprint arXiv:1810.10703* (2018).

[192] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, et al. "Parameter-efficient transfer learning for NLP". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 2790–2799.

[193] Asa Cooper Stickland and Iain Murray. "Bert and pals: Projected attention layers for efficient adaptation in multi-task learning". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 5986–5995.

[194] Sherry Yang, Yilun Du, Bo Dai, et al. "Probabilistic Adaptation of Black-Box Text-to-Video Models". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=pjtIEgscE3.

[195] Zhen Xing, Qi Dai, Han Hu, Zuxuan Wu, and Yu-Gang Jiang. "SimDA: Simple Diffusion Adapter for Efficient Video Generation". In: *arXiv preprint arXiv:2308.09710* (2023).

[196] Alexander Sax, Jeffrey Zhang, Amir Zamir, Silvio Savarese, and Jitendra Malik. "Side-Tuning: Network Adaptation via Additive Side Networks". In: *European Conference on Computer Vision*. 2020.

[197] Lvmin Zhang and Maneesh Agrawala. "Adding conditional control to text-to-image diffusion models". In: *ICCV* (2023).

[198] Zhe Chen, Yuchen Duan, Wenhai Wang, et al. "Vision Transformer Adapter for Dense Predictions". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=plKu2GByCNW.

[199] Amir Rosenfeld and John K Tsotsos. "Incremental learning through deep adaptation". In: *IEEE transactions on pattern analysis and machine intelligence* 42.3 (2018), pp. 651–663.

[200] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. "Efficient parametrization of multi-domain deep neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8119–8127.

[201] Peng Gao, Shijie Geng, Renrui Zhang, et al. "Clip-adapter: Better vision-language models with feature adapters". In: *International Journal of Computer Vision* (2023), pp. 1–15.

[202] Yi-Lin Sung, Jaemin Cho, and Mohit Bansal. "Vl-adapter: Parameter-efficient transfer learning for vision-and-language tasks". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 5227–5237.

[203] Chong Mou, Xintao Wang, Liangbin Xie, et al. "T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models". In: *arXiv preprint arXiv:2302.08453* (2023).

[204] Stephanie Fu, Netanel Tamir, Shobhita Sundaram, et al. "DreamSim: Learning New Dimensions of Human Visual Similarity using Synthetic Data". In: *arXiv preprint arXiv:2306.09344* (2023).

[205] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data". In: *Artificial intelligence and statistics*. PMLR. 2017, pp. 1273–1282.

[206] Jianyu Wang, Zachary Charles, Zheng Xu, et al. "A field guide to federated optimization". In: *arXiv preprint arXiv:2107.06917* (2021).

[207] Tao Lin, Sebastian U. Stich, Kumar Kshitij Patel, and Martin Jaggi. "Don't Use Large Mini-batches, Use Local SGD". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=B1eyO1BFPr.

[208] Daniel Povey, Xiaohui Zhang, and Sanjeev Khudanpur. "Parallel training of dnns with natural gradient and parameter averaging". In: *arXiv preprint arXiv:1410.7455* (2014).

[209] Virginia Smith, Simone Forte, Ma Chenxin, et al. "CoCoA: A general framework for communication-efficient distributed optimization". In: *Journal of Machine Learning Research* 18 (2018), p. 230.

[210] Hang Su and Haoyu Chen. "Experiments on parallel training of deep neural network using model averaging". In: *arXiv preprint arXiv:1507.01239* (2015).

[211] Jian Zhang, Christopher De Sa, Ioannis Mitliagkas, and Christopher Ré. "Parallel SGD: When does averaging help?" In: *arXiv preprint arXiv:1606.07365* (2016).

[212] Xiangru Lian, Ce Zhang, Huan Zhang, et al. "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent". In: *Advances in neural information processing systems* 30 (2017).

[213] Anastasia Koloskova, Sebastian Stich, and Martin Jaggi. "Decentralized stochastic optimization and gossip algorithms with compressed communication". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3478–3487.

[214] Anastasia Koloskova, Nicolas Loizou, Sadra Boreiri, Martin Jaggi, and Sebastian Stich. "A unified theory of decentralized sgd with changing topology and local updates". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 5381–5393.

[215] Daniel Coquelin, Charlotte Debus, Markus Götz, et al. "Accelerating neural network training with distributed asynchronous and selective optimization (DASO)". In: *Journal of Big Data* 9.1 (2022), p. 14.

[216] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. "Training deep nets with sublinear memory cost". In: *arXiv preprint arXiv:1604.06174* (2016).

[217] Aidan N Gomez, Mengye Ren, Raquel Urtasun, and Roger B Grosse. "The reversible residual network: Backpropagation without storing activations". In: *Advances in neural information processing systems* 30 (2017).

[218] Karttikeya Mangalam, Haoqi Fan, Yanghao Li, et al. "Reversible vision transformers". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 10830–10840.

[219] Yujun Lin, Song Han, Huizi Mao, Yu Wang, and Bill Dally. "Deep Gradient Compression: Reducing the Communication Bandwidth for Distributed Training". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=SkhQHMW0W.

[220] Alham Fikri Aji and Kenneth Heafield. "Sparse Communication for Distributed Gradient Descent". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Ed. by Martha Palmer, Rebecca Hwa, and Sebastian Riedel. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 440–445. DOI: 10.18653/v1/D17-1045. URL: https://aclanthology.org/D17-1045.

[221] Wei Wen, Cong Xu, Feng Yan, et al. "Terngrad: Ternary gradients to reduce communication in distributed deep learning". In: *Advances in neural information processing systems* 30 (2017).

[222] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. "On the Convergence of FedAvg on Non-IID Data". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=HJxNAnVtDS.

[223] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. "Measuring the effects of non-identical data distribution for federated visual classification". In: *arXiv preprint arXiv:1909.06335* (2019).

[224] Jianyu Wang, Vinayak Tantia, Nicolas Ballas, and Michael Rabbat. "SlowMo: Improving Communication-Efficient Distributed SGD with Slow Momentum". In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=SkxJ8REYPH.

[225] Sashank J. Reddi, Zachary Charles, Manzil Zaheer, et al. "Adaptive Federated Optimization". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=LkFG3lB13U5.

[226] Yangrui Chen, Cong Xie, Meng Ma, et al. "SAPipe: Staleness-Aware Pipeline for Data Parallel DNN Training". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 17981–17993.

[227] Binhang Yuan, Yongjun He, Jared Davis, et al. "Decentralized training of foundation models in heterogeneous environments". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 25464–25477.

[228] Jue Wang, Yucheng Lu, Binhang Yuan, et al. "CocktailSGD: Fine-tuning Foundation Models over 500Mbps Networks". In: *International Conference on Machine Learning*. PMLR. 2023, pp. 36058–36076.

[229] Arthur Douillard, Qixuan Feng, Andrei A Rusu, et al. "Diloco: Distributed low-communication training of language models". In: *arXiv preprint arXiv:2311.08105* (2023).

[230] C Daniel Freeman and Joan Bruna. "Topology and geometry of half-rectified network optimization". In: *arXiv preprint arXiv:1611.01540* (2016).

[231] Felix Draxler, Kambis Veschgini, Manfred Salmhofer, and Fred Hamprecht. "Essentially no barriers in neural network energy landscape". In: *International conference on machine learning*. PMLR. 2018, pp. 1309–1318.

[232] Stanislav Fort and Stanislaw Jastrzebski. "Large scale structure of neural network loss landscapes". In: *Advances in Neural Information Processing Systems* 32 (2019).

[233] Jonathan Frankle, Gintare Karolina Dziugaite, Daniel Roy, and Michael Carbin. "Linear mode connectivity and the lottery ticket hypothesis". In: *International Conference on Machine Learning*. PMLR. 2020, pp. 3259–3269.

[234] Johanni Brea, Berfin Simsek, Bernd Illing, and Wulfram Gerstner. "Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape". In: *arXiv preprint arXiv:1907.02911* (2019).

[235] Norman Tatro, Pin-Yu Chen, Payel Das, et al. "Optimizing mode connectivity via neuron alignment". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 15300–15311.

[236] Rahim Entezari, Hanie Sedghi, Olga Saukh, and Behnam Neyshabur. "The role of permutation invariance in linear mode connectivity of neural networks". In: *arXiv preprint arXiv:2110.06296* (2021).

[237] Berfin Simsek, François Ged, Arthur Jacot, et al. "Geometry of the loss landscape in overparameterized neural networks: Symmetries and invariances". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 9722–9732.

[238] Samuel Ainsworth, Jonathan Hayase, and Siddhartha Srinivasa. "Git Re-Basin: Merging Models modulo Permutation Symmetries". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=CQsmMYmlP5T.

[239] Mitchell Wortsman, Suchin Gururangan, Shen Li, et al. "lo-fi: distributed fine-tuning without communication". In: *arXiv preprint arXiv:2210.11948* (2022).

[240] Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. "Optimal convergence rates for convex distributed optimization in networks". In: *Journal of Machine Learning Research* 20 (2019), pp. 1–31.
[241] Liping Yi, Han Yu, Gang Wang, and Xiaoguang Liu. "FedLoRA: Model-Heterogeneous Personalized Federated Learning with LoRA Tuning". In: *arXiv preprint arXiv:2310.13283* (2023).
[242] Yaqing Wang, Subhabrata Mukherjee, Xiaodong Liu, et al. "Adamix: Mixture-of-adapter for parameter-efficient tuning of large language models". In: *arXiv preprint arXiv:2205.12410* 1.2 (2022), p. 4.
[243] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *International Conference on Learning Representations*. 2015.
[244] Ilya Loshchilov and Frank Hutter. "Decoupled Weight Decay Regularization". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Bkg6RiCqY7.
[245] Eric Wang. *alpaca-lora*. https://github.com/tloen/alpaca-lora. 2023.
[246] Tim Dettmers. *bitsandbytes*. https://github.com/TimDettmers/bitsandbytes. 2023.
[247] huggingface. *peft*. https://github.com/huggingface/peft. 2023.
[248] Prateek Yadav, Derek Tam, Leshem Choshen, Colin Raffel, and Mohit Bansal. "TIES-Merging: Resolving Interference When Merging Models". In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=xtaX3WyCj1.
[249] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, et al. "Editing models with task arithmetic". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=6t0Kwf8-jrj.
[250] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, et al. "Scaffold: Stochastic controlled averaging for federated learning". In: *International conference on machine learning*. PMLR. 2020, pp. 5132–5143.
[251] Michael S Matena and Colin A Raffel. "Merging models with fisher-weighted averaging". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 17703–17716.
[252] Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).
[253] Sebastian U. Stich. "Local SGD Converges Fast and Communicates Little". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=S1g2JnRcFX.
[254] Jianyu Wang and Gauri Joshi. "Cooperative SGD: A unified framework for the design and analysis of local-update SGD algorithms". In: *The Journal of Machine Learning Research* 22.1 (2021), pp. 9709–9758.
[255] Hao Yu, Sen Yang, and Shenghuo Zhu. "Parallel restarted SGD with faster convergence and less communication: Demystifying why model averaging works for deep learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2019.
[256] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, et al. "Mlp-mixer: An all-mlp architecture for vision". In: *Advances in neural information processing systems* 34 (2021), pp. 24261–24272.

[257] Adam Coates, Andrew Ng, and Honglak Lee. "An analysis of single-layer networks in unsupervised feature learning". In: *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2011, pp. 215–223.

[258] Gregory Griffin, Alex Holub, and Pietro Perona. "Caltech-256 object category dataset". In: (2007).

[259] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. "Sun database: Large-scale scene recognition from abbey to zoo". In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE. 2010, pp. 3485–3492.

[260] Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. "An empirical model of large-batch training". In: *arXiv preprint arXiv:1812.06162* (2018).

[261] Christopher J Shallue, Jaehoon Lee, Joseph Antognini, et al. "Measuring the effects of data parallelism on neural network training". In: *Journal of Machine Learning Research* 20.112 (2019), pp. 1–49.

[262] Samuel L. Smith and Quoc V. Le. "A Bayesian Perspective on Generalization and Stochastic Gradient Descent". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=BJij4yg0Z.

[263] Shen Li, Yanli Zhao, Rohan Varma, et al. "Pytorch distributed: Experiences on accelerating data parallel training". In: *arXiv preprint arXiv:2006.15704* (2020).

[264] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. "Measuring statistical dependence with Hilbert-Schmidt norms". In: *International conference on algorithmic learning theory*. Springer. 2005, pp. 63–77.

[265] Le Song, Alex Smola, Arthur Gretton, Justin Bedo, and Karsten Borgwardt. "Feature Selection via Dependence Maximization." In: *Journal of Machine Learning Research* 13.5 (2012).

[266] Mathilde Caron, Hugo Touvron, Ishan Misra, et al. "Emerging properties in self-supervised vision transformers". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 9650–9660.

[267] Alejandro Lopez-Cifuentes, Marcos Escudero-Vinolo, Jesus Bescos, and Alvaro Garcia-Martin. "Semantic-aware scene recognition". In: *Pattern Recognition* 102 (2020), p. 107256.

[268] Manel Baradad, Richard Chen, Jonas Wulff, et al. "Procedural image programs for representation learning". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 6450–6462.

[269] Manel Baradad, Jonas Wulff, Tongzhou Wang, Phillip Isola, and Antonio Torralba. "Learning to See by Looking at Noise". In: *Advances in Neural Information Processing Systems*. 2021.

[270] BigScience, Teven Le Scao, Angela Fan, et al. "Bloom: A 176b-parameter open-access multilingual language model". In: *arXiv preprint arXiv:2211.05100* (2022).

[271] Xinyang Geng and Hao Liu. *OpenLLaMA: An Open Reproduction of LLaMA*. May 2023. URL: https://github.com/openlm-research/open_llama.

[272] Hugo Touvron, Louis Martin, Kevin Stone, et al. "LLaMA 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* (2023).

[273] Dirk Groeneveld, Iz Beltagy, Pete Walsh, et al. "Olmo: Accelerating the science of language models". In: *arXiv preprint arXiv:2402.00838* (2024).
[274] Gemma Team, Thomas Mesnard, Cassidy Hardin, et al. "Gemma: Open models based on gemini research and technology". In: *arXiv preprint arXiv:2403.08295* (2024).
[275] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, et al. "Mistral 7B". In: *arXiv preprint arXiv:2310.06825* (2023).
[276] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, et al. "Mixtral of experts". In: *arXiv preprint arXiv:2401.04088* (2024).
[277] Thomas Wolf, Lysandre Debut, Victor Sanh, et al. "Huggingface's transformers: State-of-the-art natural language processing". In: *arXiv preprint arXiv:1910.03771* (2019).
[278] Ross Wightman. *PyTorch Image Models*. https://github.com/rwightman/pytorch-image-models. 2021.
[279] Martin Schrimpf, Jonas Kubilius, Ha Hong, et al. "Brain-score: Which artificial neural network for object recognition is most brain-like?" In: *BioRxiv* (2018), p. 407007.
[280] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 30318–30332.
[281] Antonio Torralba, Rob Fergus, and William T Freeman. "80 million tiny images: A large data set for nonparametric object and scene recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 30.11 (2008), pp. 1958–1970.
[282] Roger N Shepard. "Multidimensional scaling, tree-fitting, and clustering". In: *Science* 210.4468 (1980), pp. 390–398.
[283] Wolfgang Kabsch. "A solution for the best rotation to relate two sets of vectors". In: *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 32.5 (1976), pp. 922–923.
[284] Wolfgang Kabsch. "A discussion of the solution for the best rotation to relate two sets of vectors". In: *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography* 34.5 (1978), pp. 827–828.
[285] Shinji Umeyama. "Least-squares estimation of transformation parameters between two point patterns". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 13.04 (1991), pp. 376–380.
[286] Delwin T Lindsey and Angela M Brown. "The color lexicon of American English". In: *Journal of vision* 14.2 (2014), pp. 17–17.
[287] Roman Vershynin. *High-dimensional probability: An introduction with applications in data science*. Vol. 47. Cambridge university press, 2018.
[288] RV Mises and Hilda Pollaczek-Geiringer. "Praktische Verfahren der Gleichungsauflösung." In: *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik* 9.1 (1929), pp. 58–77.
[289] Adam Paszke, Sam Gross, Francisco Massa, et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, et al. Curran Associates, Inc., 2019, pp. 8024–8035.

[290] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. "Aggregated residual transformations for deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 1492–1500.

[291] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, pp. 448–456.

[292] Ian J Goodfellow, Oriol Vinyals, and Andrew M Saxe. "Qualitatively characterizing neural network optimization problems". In: *arXiv preprint arXiv:1412.6544* (2014).

[293] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 249–256.

[294] Jeremy Bernstein, Chris Mingard, Kevin Huang, Navid Azizan, and Yisong Yue. "Automatic Gradient Descent: Deep Learning without Hyperparameters". In: *arXiv:2304.05187* (2023).

[295] Dominic Masters and Carlo Luschi. "Revisiting small batch training for deep neural networks". In: *arXiv preprint arXiv:1804.07612* (2018).

[296] Jianxiong Xiao, Krista A Ehinger, James Hays, Antonio Torralba, and Aude Oliva. "Sun database: Exploring a large collection of scene categories". In: *International Journal of Computer Vision* 119 (2016), pp. 3–22.

[297] Andrej Karpathy. *nanoGPT*. https://github.com/karpathy/nanoGPT. 2023.

[298] Ronen Eldan and Yuanzhi Li. "TinyStories: How Small Can Language Models Be and Still Speak Coherent English?" In: *arXiv preprint arXiv:2305.07759* (2023).

[299] Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. "Feature Learning in Infinite Depth Neural Networks". In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=17pVDnpwwl.